Amatérské RADIO

ŘADA B
PRO KONSTRUKTÉRY

ČASOPIS
PRO ELEKTRONIKU
A AMATÉRSKÉ VYSÍLÁNÍ
ROČNÍK XXIX/1980 ČÍSLO 3

V TOMTO SEŠITĚ

AMATÉRSKÉ RADIO ŘADA B

Vydává ÚV Svazarmu ve vydavatelství NAŠE VOJSKO, Vladislavova 26, PSČ 113 66 Praha 1, telefon 26 06 51–7. Šéfredaktor ing. F. Smolík, redaktor L. Kalousek. Redakční rada: K. Bartoš, V. Brzák, RNDr. V. Brunnhofer, K. Donát, A. Glanc, I. Harminc, Z. Hradiský, P. Horák, J. Hudec, ing. J. T. Hyan, ing. J. Jaroš, doc. ing. dr. M. Joachim, ing. J. Klabal, ing. E. Králík, RNDr. L. Kryška, PhDr. E. Křížek, ing. E. Môcik, K. Novák, RND. L. Ondriš, ing. O. Petráček, ing. M. Smolka, doc. ing. J. Vackář, laureát st. ceny KG, ing. J. Zíma. Redakce Jungmannova 24, PSČ 113 66 Praha 1, telefon 26 06 51–7, ing. Smolík linka 354, Kalousek linka 535, sekretářka linka 355. Ročně vyjde 6 čísel. Cena výtisku 5 Kčs, pololetní předplatné 15 Kčs. Rozšiřuje PNS, v jednotkách ozbrojených sil vydavatelství NAŠE VOJSKO, administrace Vladislavova 26, Praha 1. Objednávky přijímá každá pošta i doručovatel. Objednávky do zahraničí vyřizuje PNS, vývoz tisku, Jindřišská 14, Praha 1. Tiskne NAŠE VOJSKO, n. p., závod 08, 162 00 Praha 6-Liboc, Vlastina 710.

Za původnost a správnost příspěvku ručí autor.

Návštěvy v redakci a telefonické dotazy pouze po 14. hodině.

Číslo indexu 46 044.

Toto číslo má vyjít podle plánu 15. května 1980

SVAZARM SPOLUPRÁCE ČSVTS

„V rozvoji všech radioamatérských činností bude na místě trvale prohlubovat spolupráci s Československou vědeckotechnickou společností, s Československou televizí a rozhlasem, organizacemi SSM a ROH, ministerstvem národní obrany a ministerstvem spojů."

Z dokumentu Směry a úkoly dalšího rozvoje radistické činnosti ve Svazarmu, Praha, ÚV Svazarmu 1978.

Československá vědeckotechnická společnost (ČSVTS) oslavuje v letošním roce již 25 let svého trvání. Nejprve byla ustavena při ČSAV (v roce 1955), později byla organizačně začleněna do ROH (v roce 1959) a v současné době představuje důležitou samostatnou složku Národní fronty. Jejím posláním je šířit nové vědecké a technické znalosti prostřednictvím odborných akcí (kursů, seminářů, konferencí atd.) a poradenské činnosti a urychlovat tak zavádění moderních vědeckotechnických poznatků do běžné výroby i do dalších oblastí společenské činnosti. Vedle tohoto poslání plní důležité úkoly v politické výchově vědeckotechnické inteligence. Součástmi ČSVTS jsou Česká elektrotechnická společnost (adresa jejího ústředního výboru: 110 01 Praha 1, Široká 5) a Slovenská elektrotechnická společnost (adresa ústředního výboru: 898 17 Bratislava, Kocelova 15), jejichž činnost bude asi zajímat čtenáře AR nejvíce. ČÚV Elektrotechnické společnosti ČSVTS ustavil odborné orgány pro tyto dílčí specializované oblasti elektrotechniky (tzv. ústřední odborné skupiny – ÚOS):

1. Projektování a montáž elektrických zařízení
2. Elektrické přístroje a rozváděče nn, vn a elektroinstalační materiál
3. Elektrické přístroje vvn
4. Transformátory
5. Elektrické stroje točivé do 1 MW
6. Elektrické pohony
7. Výkonová elektronika
8. Provoz, údržba a revize elektrických zařízení
9. Aktivní prvky v elektronice
10. Součástky pro elektroniku
11. Modulární elektrické systémy
12. Mikroprocesorová technika
13. Elektrické měřicí přístroje
14. Diagnostika v elektronice
15. Biomedicínské inženýrství
16. Mikrovlny a optoelektronika
17. Telekomunikace
18. Statická elektřina
19. Nové přeměny energie
20. Robotika
21. Elektrochemie
22. Technická normalizace
23. Historie elektrotechniky a elektroniky

Uvádíme úplný výčet ÚOS, protože obsah radioamatérské zájmové činnosti se neustále rozšiřuje a v současné době pokrývá prakticky celý obor elektrotechniky. Oblast odborného zájmu radioamatérských organizací Svazarmu a Elektrotechnické společnosti ČSVTS je si tedy velmi blízká; radioamatérská činnost je zaměřena více na praktickou stránku radiotechniky a elektroniky, zatímco práce Elektrotechnické společnosti ČSVTS se zakládá na hlubokých teoretických znalostech elektrotechniky a nelze ji označit přívlastkem amatérská. Mnozí svazarmovští radioamatéři jsou však současně i členy ČSVTS.

Spolupráce partnerů s takto vzájemně se doplňujícími vlastnostmi bude na první pohled výhodná, a proto nejvyšší orgány Svazarmu i ČSVTS mají na ní zájem, i když oficiální dohoda o vzájemné spolupráci v oboru radiotechniky a elektrotechniky zatím učiněna nebyla. Podle názoru čelních představitelů radioamatérské organizace ve Svazarmu i Elektrotechnické společnosti ČSVTS bude nejdůležitějším bodem spolupráce výměna odborných informací – formální i neformální. ČSVTS má velmi bohatou ediční činnost, kterou představuje množství sborníků ze seminářů, konferencí a dalších akcí, pořádaných ČSVTS. Mnohé informace v nich obsažené mohou využít pro svoje potřeby i radioamatéři. Na druhé straně informace a zkušenosti z radioamatérské činnosti přinášejí časopisy AR a RZ a některé monografie, i když zatím v dosti omezeném množství. Sborníky ze seminářů techniky a provozu na KV a VKV jsou rovněž pravidelně vydávány.

Bude třeba zlepšit vzájemnou informovanost členů obou organizací o pořádání a obsahu jednotlivých akcí. (Účast na akcích ČSVTS není podmíněna členstvím v této organizaci a je tedy dána hlavně zájmem a časovými a finančními možnostmi radioamatérů.) Tam je totiž optimální prostředí pro osobní výměnu zkušeností a informací.

V současné době je aktuální otázkou, diskutovanou mezi radioamatéry, ustavování radiotechnických kabinetů při KV a OV Svazarmu. V návrhu na směrnici pro činnost metodických radiotechnických kabinetů, vypracovaném ÚRRA Svazarmu, stojí hned v úvodu: „Radiotechnické kabinety pomáhají územním orgánům organizovat a provádět odborné kursy radiotechniky, rádiového provozu, speciální kursy měřicí, televizní, polovodičové techniky, automatizace a jiné kursy, které budou sloužit pro členy i nečleny Svazarmu podle místních požadavků závodů, zájmu občanů a podle technických a kádrových podmínek". V kapitole o řízení radiotechnických kabinetů se říká: „Radiotechnické kabinety řídí odborné komise kabinetů, schválené příslušnými krajskými nebo okresními výbory a na návrh rady odbornosti. Krajské i okresní výbory získávají za členy odborné komise kabinetu politicky a odborně vyspělé pracovníky slaboproudého průmyslu, inženýry, profesory vysokých a průmyslových škol, pedagogické pracovníky, jakož i techniky a provozně vyspělé radioamatéry".

Zabezpečení dobrého chodu radiotechnických kabinetů přímo nabízí další možnosti spolupráce s ČSVTS, která jistě v souladu se svým posláním ochotně poskytne z řad svých členů lektory pro radiotechnické kabinety. Lektorská práce členů ČSVTS pro potřeby Svazarmu se již v praxi osvědčuje – např. na přehlídkách Hifi-Ama nebo zatím méně často při pořádání odborných přednášek v jednotlivých ZO Svazarmu.

Nedávno referoval náš časopis (AR A1/80) o zřizování vysokoškolských rad Svazarmu. Na vysokých školách technického zaměření je činnost ČSVTS značně rozšířená – např. fakultní pobočka ČSVTS na fakultě elektrotechnické ČVUT v Praze pořádá sama (ale také ve spolupráci s SSM) různé akce výukového charakteru a studentské vědecké konference, které probíhají v něko-

lika specializovaných sekcích, mezi nimiž je i sekce radiotechnická. Vysokoškolské rady Svazarmu mají možnost být iniciátorem spolupráce svazarmovských radioamatérů s pobočkami ČSVTS na vysokých školách.

Na závěr vás seznámíme s některými akcemi ČSVTS, které proběhnou ve druhé polovině letošního roku a svým obsahem jsou pro radioamatéry zajímavé. Uvádíme výběr z celostátních a republikových akcí, ale věnujte pozornost i činnosti krajských, závodních nebo fakultních rad ČSVTS. V závorce za názvem akce je vždy uvedeno označení akce, čtvrtletí, druh akce, místo konání a adresa, na niž je možno se přihlásit nebo získat další informace:

- *Hybridní integrované obvody* (133bC, IV., konference, Pardubice, Dům techniky ČSVTS, 532 27 Pardubice, tř. Míru 113)
- *Kurs malé výpočetní techniky pro elektrotechniky* (136C, III., kurs, Praha, ČVUT, fakulta elektrotechnická, 160 00 Praha 6, Suchbátarova 2)
- *Programovací jazyk PASCAL* (140C, III., kurs, Praha, ČVUT, fakulta elektrotechnická, 160 00 Praha 6, Suchbátarova 2)
- *Mikroprocesory a jejich využití* (141C, III., kurs, Praha, ČVUT, fakulta elektrotechnická, 160 00 Praha 6, Suchbátarova 2)
- *Plošné spoje* (146C, III., konference, Pardubice, Dům techniky ČSVTS, 532 27 Pardubice, tř. Míru 113)
- *Polovodiče v praxi* (147C, IV., seminář, Tábor, Dům techniky ČSVTS, 370 21 České Budějovice, 5. května 42)
- *Metody projektování zařízení s integrovanými obvody* (148C, IV., seminář, Pardubice, Dům techniky ČSVTS, 532 27 Pardubice, tř. Míru 113)
- *Mikroprocesorová technika* (149C, IV., seminář, Ostrava, Dům techniky ČSVTS, 709 00 Ostrava-Mariánské Hory, Dr. Maye 6)
- *Elektrotechnika v zemědělství* (152C, IV., konference, České Budějovice, Dům techniky ČSVTS, 370 21 České Budějovice, 5. května 42)
- *Obvody a tyristory* (154C, IV., kurs, Beskydy, Dům techniky ČSVTS, 709 00 Ostrava-Mariánské Hory, Dr. Maye 6).

Za podklady a informace pro tento článek děkujeme ing. Zdeňku Proškovi, tajemníkovi ČÚV Elektrotechnické společnosti ČSVTS a doc. Jiřímu Vackářovi, CSc., vědeckému tajemníkovi ČSVTS pro slaboproudou elektrotechniku. *pfm*

PRAXE ČÍSLICOVÉ TECHNIKY

Miroslav Háša

Úvod

Jistě jste si, vážení čtenáři, všimli, že nebývalý rozmach elektroniky v oblasti číslicové techniky se projevuje jak v obsahu zájmové radioamatérské činnosti, tak i v obsahu časopisu AR. Tato nová zájmová oblast zasahuje v současné době do oboru automatizace a kybernetiky a stává se zájmovou technickou činností mnoha amatérů i profesionálů. Přitom znalosti o číslicové technice, samočinných počítačích a jejich programování jsou přinejmenším stejně přístupné, jako znalosti z oblasti klasické elektrotechniky a radiotechniky.

Proto jsme v oddělení elektroniky a kybernetiky Městské stanice mladých techniků při Domu pionýrů a mládeže Hlavního města Prahy začali pracovat v zájmových kroužcích s logickými integrovanými obvody. Přístup dětí, jejich zájem a schopnosti nás překvapily. Oddělení začínají navštěvovat děti ve věku 9 až 10 let. Poznávají či opakují si dvojkovou početní soustavu, výrokovou logiku a základní logické funkce. Děti poznávají princip černé schránky a učí se poznávat funkce jednotlivých integrovaných obvodů, princip samočinného počítače von Neumannova typu a základy programování.

Teoretické základy klasické elektroniky, jako Ohmův zákon, Kirchhoffovy zákony atd. jsou důležité při obvodových řešeních a technických aplikacích polovodičových součástek, proto jejich výklad považujeme v začátcích za ztrátu času a děti se s nimi seznamují v kroužku až tehdy, když je přímo potřebují (např. s funkcí tranzistoru se seznamují při výkladu činnosti hradla s otevřeným kolektorem). V průběhu vysvětlování základů číslicové techniky pracují děti se stavebnicemi. Je sice pravda, že neumí navrhnout jednoduchý zesilovač s tranzistory, jsou však na druhé straně schopny realizovat své jednoduché návrhy formou skládání „černých skříněk".

Členové zájmových kroužků se podíleli i na návrhu a realizaci učebních pomůcek. Během tří let byl v oddělení vytvořen soubor vyučovacích pomůcek, který převážně zábavnou a populární formou seznamuje zájemce s číslicovou technikou, se základy samočinných počítačů a směřuje k pochopení základů kybernetiky (viz fotografie na 2. str. obálky).

Do tohoto souboru patří např. stavebnice Minilogik (AR řady A, č. 12/78), papírový počítač PP 78 (VTM č. 15 až 18/79), papírový počítač PP 79 (seriál v časopisu ABC), Computer kufr 79, modulové stavebnice Dominologik a Laboratoř číslicové techniky, programovatelný instrukční procesor PIP-2 atd. Mimo uvedené pomůcky jsou v oddělení připravovány audiovizuální pořady (diafon, obrazový záznam, film) a tematicky zaměřené texty.

Předkládáme čtenářům AR část těchto textů a byli bychom rádi, kdyby pomohly zpřístupnit číslicovou techniku tomu, kdo o ni projeví zájem.

V první části najdete úvod do problematiky mikroprocesorů. Jsou v něm stručně probrány základy číslicové techniky a výklad je zakončen popisem činnosti instrukčního čtyřbitového programovatelného procesoru PIP-2. Pochopení činnosti (a případná stavba tohoto procesoru) může být pro mladé adepty číslicové techniky velmi dobrou průpravou nejen k práci s mikroprocesory a mikropočítači, ale poskytuje i cenné informace pro zájemce, kteří by se chtěli pokusit o samostatnou konstrukci podobných zařízení.

Protože (vzhledem k počtu stránek, který byl k dispozici a vzhledem k tomu, že v AR již vyšla škola základů číslicové techniky) jde o stručné shrnutí problémů z číslicové techniky, doporučujeme tuto část AR řady B především těm, kteří mají již alespoň drobné zkušenosti se zapojováním integrovaných obvodů (např. ve stavebnicích).

Konstrukce a činnost zařízení s logickými obvody jsou obvykle vysvětlovány jinou formou, než jaká je použita v závěrečné Kuchařce TTL a v článku Programátor ústředního topení. Snahou bylo přiblížit čtenáři co nejvíce postup při návrhu a realizaci zařízení využívajícího číslicové techniky, objasnit přístup k zadanému úkolu a naznačit kromě řešení běžnými logickými obvody TTL i řešení s mikroprocesorem typu 8080, který je zařazen do výrobního programu n. p. TESLA i dalších výrobců v socialistických státech. Programátor ústředního topení byl zvolen záměrně jako ukázka toho, jak by elektronika mohla přispět k hospodaření s energií.

Číslicová technika, logické integrované obvody, samočinné počítače a jejich programování – to vše se zdá být mnohem složitější než elektrotechnika minulých let. Začnete-li však pracovat s mládeží, brzičko vás děti svými znalostmi přesvědčí, že pravý opak je pravdou. Navíc je dnes již nesporné, že číslicová technika je nejperspektivnější oblastí elektroniky, neboť její možnosti jsou z dnešního pohledu téměř nevyčerpatelné.

Radiotechnika a elektronika měla v minulosti v Československu vysokou úroveň, je však chybou, že československý průmysl a výzkumná a vývojová základna nezachytila současný nebývalý rozvoj elektroniky a číslicové techniky včas – a tak nás jistě bude stát velmi mnoho sil po několik desítek let, budeme-li se chtít vyrovnat vyspělému elektronickému průmyslu těch států, které tento trend rozpoznaly včas a včas reagovaly na příslušné prognózy.

Chtít, začít a mít dobrou vůli je počátkem každého řešení nahromaděných problémů. Proto vychází redakce AR vstříc snahám pracovníků oddělení elektroniky a kybernetiky Městské stanice mladých techniků při Domu pionýrů a mládeže Hlavního města Prahy a uveřejňuje to, co se v praxi ukázalo jako velmi potřebné a osvědčené. V MSMT se již čtvrtým rokem scházejí s nejrůznějšími nápady a návrhy (např. jak nahradit přídavná zařízení mikropočítačů, která nejsou zatím u nás k dispozici – jak připojit kazetový magnetofon jako externí paměť počítače, jak realizovat hexadecimální klávesnici, jak udělat grafický displej z televizoru, jak realizovat komunikaci mezi počítači atd.), diskutují o řešení těchto otázek se znalostí programové i obvodové problematiky.

Ti nejstarší využívají možnosti pracovat o prázdninách ve vývojových a výzkumných ústavech, kde sbírají zkušenosti přímo z praxe. Součástí této zájmové činnosti je i studium domácí a zahraniční literatury, nad níž se vede mnoho diskusí i sporů, z nichž se líhnou nové plány do budoucnosti.

Výsledky této zájmové činnosti se projevily i určitým veřejným uznáním na celostátní výstavě STTM 1979 v Olomouci, kde ti starší

získali několik cen. Mladší členové kroužků však vyšli se svými kybernetickými hračkami a stavebnicemi naprázdno; podle mého názoru dala porota ne zcela správně přednost „klasickým" výrobkům, které nepřinášejí většinou nic nového. Mladí se však nenechali odradit – místo pevně naprogramovaného „želva" (mužská část želví rodiny) staví nového s možností programování a v plánu mají zařízení, které se pohybuje po místnosti a je řízeno robotem, který by se učil a uměl mluvit.

Přes potíže, které v tomto oboru u nás existují, přemýšlejí členové kroužků o nejrůznějších hrách se samočinnými počítači, o trenažérech raket, letadel a kosmických lodí. *Učí se pracovat v týmu a vést kolektiv.*

Ze všeho, co jsem uvedl, vyplývá jednoznačný závěr: budeme-li hledat budoucnost československé elektroniky, musíme mnohem více pozornosti věnovat především mládeži, nezatížené technickým konzervatismem, a vést ji k týmové i samostatné práci a poskytnout ji k ní co nejlepší podmínky.

Od jednoduchých logických obvodů k mikropočítačům

Miroslav Háša

V tomto úvodním článku vás chceme seznámit nejen se základy číslicové techniky až po organizaci mikroprocesoru, ale i s návodem, jak si postavit čtyřbitový instrukční procesor, který v podstatě vysvětluje organizaci a funkci mikroprocesorů. Součástky, které jsou při stavbě použity, jsou vyráběny v zemích RVHP.

ČÍSELNÉ SYSTÉMY

Mikroprocesory již našly své místo v nové éře elektroniky. Stejně jako kdysi tranzistory „dobyly" oblasti, v nichž se používaly výlučně elektronky, tak v mnoha aplikacích vystřídaly integrované obvody tranzistory. V současné době používané mikroprocesory nahradí desítky a nebo dokonce stovky integrovaných obvodů.

Konvenční číslicový logický obvod je pevně „zadrátován" a jeho činnost nelze snadno změnit. Mikroprocesor, který je funkčně rovnocenný základní jednotce samočinného počítače, můžeme však vnějšími zásahy naprogramovat tak, aby pracoval jako řadič terminálu, kalkulátor, samočinný počítač apod. Pouhá výměna instrukcí v paměti a dodání nových instrukcí činnost mikroprocesoru zcela změní.

Mikroprocesory jsou natolik nové a zvláštní součástky, že mnozí z těch, kteří se zajímají o elektroniku, se ještě pořádně neseznámili s jejich základními operačními principy a jejich programováním. Proto v úvodu tohoto čísla AR řady B stručně osvětlíme to, co vede k pochopení činnosti mikroprocesoru a uvedeme podrobný popis architektury a činnosti jednoduchého vyučovacího mikroprocesoru PIP-2 včetně technického návrhu, který vypracoval šestnáctiletý student gymnázia Jan Mercl.

Ty nejjednodušší číslicové logické prvky pracují na základě přítomnosti nebo nepřítomnosti elektrického signálu. Tohoto faktu lze využít k reprezentaci čísel a k operacím s nimi v binárním, dvoučíslicovém systému. Proto se nejprve podíváme na základy binární a dalších číselných soustav.

Dvojkový versus desítkový systém

Desítková číselná soustava se dá snadno naučit a používat. Aspoň se to tak většina z nás učila ve škole. Ale zamyslete se na chvilku nad desítkovou aritmetikou. Abyste mohli sečíst jakákoli dvě desítková čísla, *musíte se nejprve naučit nazpaměť sto dalších* vztahů. Jaké jsou to vztahy? Jsou to vztahy číselné, jako $1 + 1 = 2$; $4 + 5 = 9$; $3 + 7 = 10$; atd. Že je to jednoduché? Ano, avšak pouze proto, že jsme se to předtím naučili.

Vidíte, že „jednoduchý" desítkový číselný systém není vůbec tak jednoduchý. A to jsme se nezmínili o pravidlech při odečítání, násobení a dělení desítkových čísel. Zkrátka, pro provádění různých operací v desítkové soustavě existují doslova stovky pravidel.

Trvalo nám pět nebo šest let, než jsme zvládli pravidla desítkové aritmetiky, ale pravidla dvojkové (binární) aritmetiky můžete zvládnout za pět nebo šest minut! Dvojková soustava má dvě číslice neboli bity, 0 a 1, takže pro provádění početních úkonů ve dvojkové aritmetice potřebujeme pouze několik málo pravidel.

Zde jsou, například, pravidla pro dvojkové sečítání:

$$0 + 0 = 0,$$
$$0 + 1 = 1,$$
$$1 + 0 = 1,$$
$$1 + 1 = 0.$$

přenos do dalšího řádu 1, nebo chcete-li 10;

$$1 + 1 + 1 = 10 + 1 = 11.$$

Těchto pět pravidel můžete použít ke sčítání dvou libovolných dvojkových čísel. Stejně jednoduchá jsou pravidla pro dvojkové odečítání. A protože násobení a dělení můžeme považovat za opakované sečítání a odečítání, jsou pravidla pro dvojkovou aritmetiku mnohem jednodušší, než pro aritmetiku desítkovou.

Pravidla pro dvojkové sečítání můžeme použít pro počítání ve dvojkové soustavě vůbec. Začnete s 0, přičtete 1 a pokračujete s přičítáním 1 vždy k následujícímu číslu. Tato procedura se nazývá „přirůstání", inkrementace, a dovoluje rychle vytvořit například prvních šestnáct dvojkových čísel:

0 100 1000 1100
1 101 1001 1101
10 110 1010 1110
11 111 1011 1111

Počítačoví specialisté se často zmiňují o dvojkových číslech jako o slovech nebo obrazcích bitů, protože se jich často používá k tomu, aby zastupovaly počítačové instrukce a nečíselné funkce. Běžně se používají slova o osmi bitech – říká se jim byty (čti bajty). Slovo, které má čtyři bity, se nazývá nibble.

I když se dá dvojková aritmetika snadno naučit, je hlavní nevýhoda dvojkové soustavy v tom, že dvojková čísla (slova) jsou často dlouhá a těžko se s nimi pracuje, těžko se pamatují, snadno se v nich dělají chyby a nesnadno se čtou. Například nějaké desítkové číslo, které se skládá pouze z jedné nebo ze dvou číslic, bude potřebovat pro své vyjádření ve dvojkové soustavě až sedm bitů. Desítkové číslo 99 se snadno čte a pamatuje. Jeho dvojkový protějšek je hrozný: 1100011.

Počítačoví nadšenci vymysleli několik šikovných zkratek a triků pro zapamatování dvojkových čísel a jejich převod do desítkové soustavy. Tyto metody se pomalu stávají pro *mikroprocesorovou generaci téměř druhou* přirozeností a tak se na ně trochu podívejme.

Převod dvojkových čísel na desítková

Převod zvládnete snadno, budete-li vědět, *jak rozložit běžné desítkové číslo na jeho* jednotlivé části. Tak např.: 653 je $600 + 50 + 3$. Pozice každé číslice v čísle jako je 653 určuje, kolikanásobkem deseti ta či ona číslice bude. Tak:

$$653 = 6 \times 10^2 = 600$$
$$5 \times 10^1 = 50$$
$$3 \times 10^0 = 3$$
$$653$$

Dvojková čísla mohou být rozšířena použitím téže metody – a pak zase přeměněna na jejich desítkové protějšky. Dvojková soustava má pouze dva bity a pozice bitu ve dvojkovém čísle je určena mocninou čísla dvě.

Tak:

$$1001 = 1 \times 2^3 = 1000$$
$$0 \times 2^2 = 0000$$
$$0 \times 2^1 = 0000$$
$$1 \times 2^0 = 0001$$
$$1001$$

Chceme-li přeměnit binární číslo 1001 na jeho desítkový ekvivalent, změníme mocniny dvojek na jejich desítkové hodnoty a sečteme výsledek:

$$1001 = 1 \times 8 = 8$$
$$0 \times 4 = 0$$
$$0 \times 2 = 0$$
$$1 \times 1 = 1$$
$$9$$

Ještě rychlejší je změnit nějaké dvojkové číslo na desítkové tak, že sestavíme vzestupnou řadu mocnin dvou z každého bitu v tomto čísle, nejméně významným bitem počínaje. Pak sečteme mocniny dvou nad každým bitem a nevšímáme si těch, které jsou nulovými bity. Takto se např. převádí dvojkové číslo 1100110 na desítkové:

64 32 16 8 4 2 1
1 1 0 0 1 1 0
$64 \times 32 + 0 + 0 + 4 + 2 + 0 = 102$

Převod desítkových čísel na dvojková

Rychlý způsob, jak převést desítkové číslo na jeho dvojkový protějšek, je dělit opakovaně desítkové číslo dvěma. Zbytky každého dělení, které budou vždy 0 nebo 1, se stanou tímto dvojkovým číslem. Přeměňme třeba číslo 102 na dvojkové při použití této metody:

$102 : 2 = 51$, zbytek 0
$51 : 2 = 25$, zbytek 1
$25 : 2 = 12$, zbytek 1
$12 : 2 = 6$, zbytek 0
$6 : 2 = 3$, zbytek 0
$3 : 2 = 1$, zbytek 1
Poslední zbytek 1

$$102 = 1\ 1\ 0\ 0\ 1\ 1\ 0$$

Osmičková a šestnáctková čísla

Dvojkových čísel se často využívá k tomu aby zastupovala počítačové instrukce a operace. Například 01110110 je dvojkový ekvi-

valent k číslu 118; 01110110 je však také kód instrukce zvolený konstruktéry firmy Intel k tomu, aby zastupovala instrukci HLT (halt = zastav) pro mikroprocesor 8080. Dvojková čísla se také používají k tomu, aby zastupovala adresy paměti uvnitř počítače. Tak 01110110 může zastupovat desítkové číslo 118, instrukci HLT nebo 119. adresu v paměti mikropočítače (přičemž první adresa je 00000000).

Protože dvojková čísla hrají důležitou roli u mikroprocesorů a počítačů, zmíníme se i o výhodných zkratkách, které šetří prostor i čas a nazývají se osmičkové a šestnáctkové číselné soustavy.

Desítková čísla mají za základ desítku, takže nejvyšší desítkovou číslicí je číslo 9. Osmičková čísla mají za základ osmičku, nejvyšší osmičkovou číslicí je číslo 7. Protože dvojkový ekvivalent desítkové číslice 7 (která je ekvivalentní osmičkové číslici 7) je 111, je snadné přeměnit jakékoli dvojkové číslo na jeho osmičkový protějšek a to tak, že rozdělíme bity v tomto čísle do skupin po třech bitech a přeměníme každou skupinu na desítkový ekvivalent. Dvojkové číslo 01110110 má pak tvar 01 110 110, neboli 166 v soustavě osmičkové.

Uvádíme-li výčet čísel, která mají rozdílné základy, je dobrým zvykem označit každé číslo indexem základu. Tudíž 166_8 je osmičkové číslo. Je jasné, že 166_8 se snáze pamatuje než 01110110_2. A je, mimochodem, velmi snadné převést 166_8 zpátky na dvojkové číslo jednoduše tak, že vypíšeme dvojkové ekvivalenty každé číslice:

$$\begin{array}{l} 1 = 01 \\ 6 = \quad 110 \\ \underline{6 = \quad 110} \\ \quad 01110110 \end{array}$$

Šestnáctková čísla mají za základ šestnáctku. Běžně se používají k tomu, aby zjednodušila 8bitové byty na snadno zapamatovatelná dvoumístná čísla.

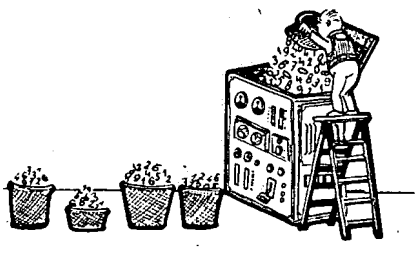
Šestnáctkové číslice jsou 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, a F. Nenechte se mýlit písmeny A až F. Desítkových číslic je více než dost pro dvojkové a osmičkové systémy, ale není jich dost pro šestnáctkové číslice. Písmena A až F doplňují šest číslic za číslicemi 0 až 9.

Je snadné přeměnit dvojkový byte na šestnáctkové číslo (zkráceně hex). Nejdřív rozdělíme byte na dva nibbly. Pak přidělíme každému nibblu ekvivalent hex. 1111_2 je F_{16} a 0110_2 je 6_{16}. Takže 11110110_2 je $F6_{16}$.

Abychom přeměnili číslo hex na dvojkové stačí, přidělíme-li dvojkový ekvivalent každé číslici hex. Takže $F6_{16}$ je 1111_2 a 0110_2, neboli 11110110_2. I když je správné rozlišovat číslo hex indexem 16, není to nezbytné v těch případech, kdy se v čísle objevují k doplnění číslic písmena.

Většina dnešních mikroprocesorů používá 8bitové adresy a slova instrukce, takže často budete vidět programy uváděné v osmičkové nebo šestnáctkové soustavě. Nějakou dobu to trvá, než si na tyto soustavy – zvláště na šestnáctkovou – zvyknete, ale potom uvidíte, že jsou velmi výhodné, zvláště u mikroprocesorů. Také vám pomůže porovnávací tabulka:

soustava			
desítková	dvojková	osmičková	šestnáctková
0	0	0	0
1	1	1	1
2	10	2	2
3	11	3	3
4	100	4	4
5	101	5	5
6	110	6	6
7	111	7	7
8	1000	10	8
9	1001	11	9
10	1010	12	A
11	1011	13	B
12	1100	14	C
13	1101	15	D
14	1110	16	E
15	1111	17	F

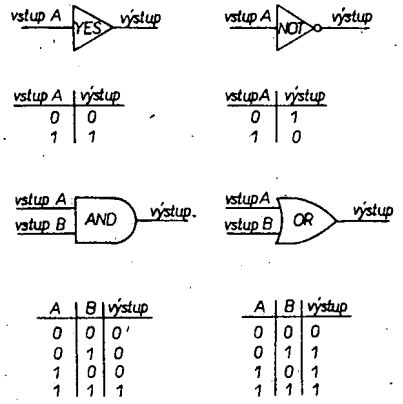


ZÁKLADNÍ LOGICKÁ HRADLA

Všechny číslicové logické obvody, od nejjednodušších čítačů k nejsložitějším mikroprocesorům, jsou vytvořeny v podstatě ze vzájemně propojených kombinací jednoduchých obvodů, nazývaných logická hradla. Existují čtyři základní logická hradla, která jsou označována podle funkce jako obvody YES, NOT, AND a OR (ANO, NE, I a NEBO). Každé z těchto základních hradel má jeden nebo několik vstupů, jednoduchý výstup a dvojici vývodů pro napájení.

Různé kombinace dvojkových bitů 0 a 1 mohou být přivedeny na vstupy nějakého hradla tak, že malé napětí bude představovat úroveň logické 0 a větší napětí (například 5 V) úroveň logické 1 – takto pracuje pozitivní logika, v negativní logice je tomu obráceně.

Hradlo YES (obr. 1) přenáší logický stav (0 nebo 1) ze vstupu přímo na výstup. Často se ho využívá k přizpůsobování logických obvodů, které jsou jinak elektronicky neslučitelné (jako oddělovač, převodník úrovně apod.).



Obr. 1. Čtyři základní logická hradla YES, NOT, AND a OR (někdy označovaná česky jako ANO, NE, I a NEBO) a jejich pravdivostní tabulky

Hradlo NOT (NE – negace) obrací nebo doplňuje logický stav svého jednoduchého vstupu, takže se často nazývá převodník nebo invertor. Funkce NOT je často indikována nějakou značkou nad symbolem pro vstup nebo výstup, který byl obrácen (invertován). Je-li např. na vstupu A úroveň log. 0 a na jiném místě, třeba B, je log. 1, pak $A = \overline{B}$ ($\overline{B}$ se čte a někdy i píše jako B NON).

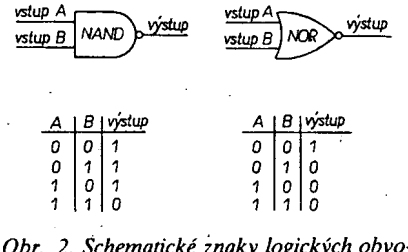
Hradlo AND (I – logický součin, disjunkce) je rozhodovací obvod se dvěma nebo více vstupy. Na jeho výstupu je log. 0, je-li na některém nebo na všech vstupech (vstup A a B a C . . .) úroveň log. 0.

Hradlo OR (NEBO – logický součet, konjunkce) je také rozhodovací obvod se dvěma nebo několika vstupy. Na jeho výstupu je log. 0, není-li na některém nebo na všech jeho vstupech (vstup A nebo B nebo C . . .) úroveň log. 1.

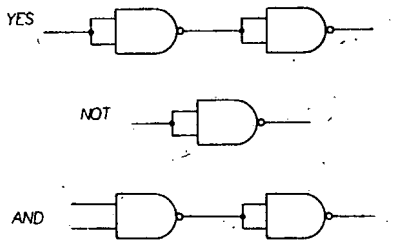
Činnost hradla může být určena tabulkou, která ukazuje kombinace vstupních bitů, jimž odpovídá určitý výstupní bit. Taková tabulka se nazývá pravdivostní tabulka (obr. 1).

Složené logické obvody

Kombinací dvou nebo několika základních hradel lze získat některé velmi důležité logické operace. Těmi dvěma nejdůležitějšími jsou NAND (negovaný logický součin) a NOR (negovaný logický součet). Symboly příslušných logických obvodů a pravdivostní tabulky jsou na obr. 2.

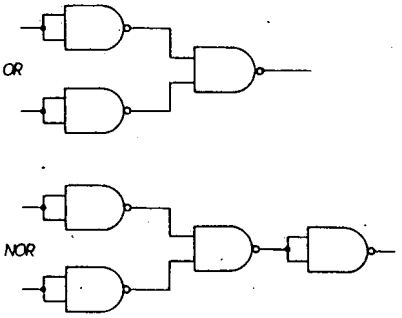


Obr. 2. Schematické znaky logických obvodů NAND a NOR s jejich pravdivostními tabulkami



Obr. 3. Použití hradel NAND k simulování logických funkcí YES, NOT a AND

Jak je zřejmé z obr. 3, mohou různé kombinace hradel NAND (nebo NOR) simulovat činnost obvodů logické negace, logického součtu a logického součinu. To je sice velmi důležité, nejvíce fascinující je však logická ekvivalence funkcí NAND a NOR. Díky pravidlu, které je známo jako DeMorganův teorém, je pozitivní NAND ekvivalentní negativnímu NOR a naopak. Můžete si to sami ověřit tím, že si napíšete příslušné pravdivostní tabulky – shledáte, že jsou totožné. DeMorganův teorém zjednodušuje číslicovou techniku do té míry, že s hradly



Obr. 4. Použití hradel NAND k modelování logických funkcí OR a NOR s využitím DeMorganova teorému

NAND nebo NOR lze realizovat jakoukoli logickou funkci.

Na obr. 4 například vidíte, jak lze z hradel NAND získat obvod jak s logickou funkcí OR, tak NOR. Povšimněte si, jak se hradel NAND využívá jako invertorů pro přechod z pozitivní na negativní logiku.

SLOŽITĚJŠÍ LOGICKÉ SYSTÉMY

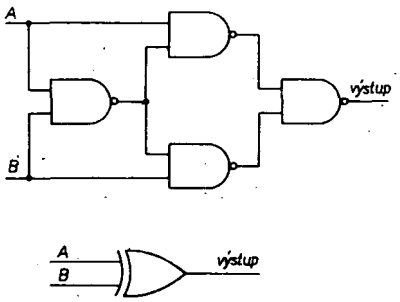
Jednoduchá a složená hradla lze vzájemně spojovat, abychom získali nejrůznější logické funkce. Některé z výsledných logických systémů obsahují hradel pouze několik; jiné mohou používat desítky nebo stovky hradel. Logické systémy mohou být rozděleny na dva základní druhy, na kombinační a sekvenční.

Kombinační obvody jsou charakterizovány rychlou provozuschopností, mají mimořádně krátké zpoždění, logická funkce je realizována v co nejkratším čase.

Sekvenční obvody obsahují paměťové a zpožďovací prvky, které dovolují, aby logický výsledek předchozího vstupu přímo ovlivnil nový vstup. Následkem toho jsou sekvenční obvody pomalejší než obvody kombinační, což umožňuje použít je jako např. paměťové registry, čítače, děliče, třídiče a mikroprocesory.

KOMBINAČNÍ LOGICKÉ OBVODY

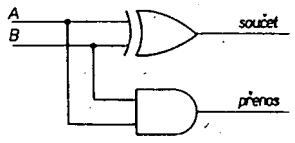
Nejjednodušším kombinačním logickým obvodem je obvod EXCLUSIVE – OR. Symbol a pravdivostní tabulka pro tento obvod jsou na obr. 5.



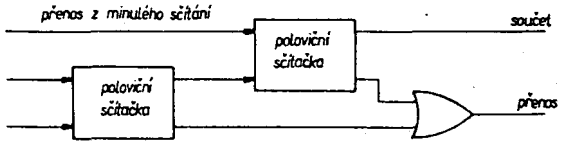
A	B	výstup
0	0	0
0	1	1
1	0	1
1	1	0

Obr. 5. Kombinační logický obvod EXCLUSIVE-OR, sestavený z hradel NAND, jeho schematický znak a pravdivostní tabulka

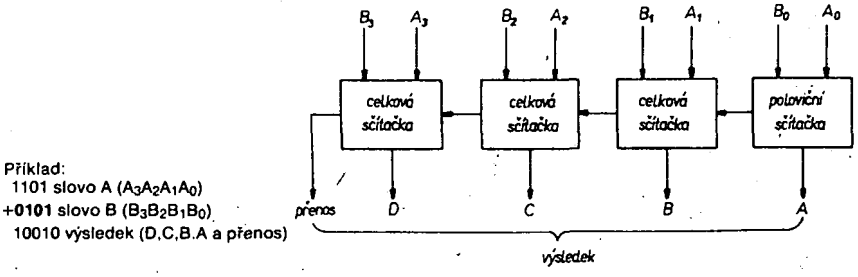
Podívejte se na chvilku na pravdivostní tabulku na obr. 5. Na výstupu je úroveň log. 1 pouze tehdy, je-li na prvním nebo druhém vstupu úroveň log. 1. V ostatních dvou případech je na výstupu úroveň log. 0. To je stejné jako u zákonů pro sčítání ve dvojkové soustavě s tou výjimkou, že bychom potřebovali při sčítání 1 + 1 získat ještě jeden bit s úrovní log. 1 jako přenos do dalšího řádu. Jak ukazuje obr. 6, můžeme takový obvod



Obr. 6. Poloviční sčítačka umožňuje sečíst dvě čísla (dva znaky) dvojkové početní soustavy



Obr. 7. Celková sčítačka, vytvořená ze dvou poloviční sčítaček a obvodu OR



Obr. 8. Řazení sčítaček pro vytvoření sčítačky, která umí sečíst dvě čtyřbitová čísla (integrovaný obvod MH7483)

vytvořit velice snadno. Tento obvod, pomocí něhož můžeme přičíst jakýkoli ze dvou dvojkových bitů, se nazývá poloviční sčítačka.

Jak jistě tušíte, je tato poloviční sčítačka velice užitečná, i když může „přijímat" pouze dva vstupní bity. K tomu, abychom doplnili zákony dvojkového (binárního) sčítání, potřebujeme sčítací obvod, který takto pracuje; je nazýván celková sčítačka a je na obr. 7. Tuto úplnou sčítačku vytvoříme ze dvou poloviční sčítaček a obvodu OR.

Ze sčítaček je možno vytvořit řetězec, tzv. dvojkovou sčítačku, která je schopna sčítat dvojková čísla zakódovaná mnohonásobnými bity. Na obr. 8 je čtyřbitová sčítačka, která dovede sčítat dvě čtyřbitová čísla, která přivedeme na vstupy. Pokuste se sečíst 1101 + 0101 s použitím této sčítačky, abyste si sami dokázali, že skutečně sčítá.

Dvojková sčítačka tvoří část aritmetickologické jednotky (ALU) mikroprocesoru. Aritmetickologická jednotka je část mikroprocesoru, která realizuje sčítání a různé logické operace se dvěma vstupujícími binárními slovy. Později si povíme o tom, jak je organizována činnost ALU, aby podle binárních stavů přivedených na řídicí vstupy prováděla aritmetické a logické operace.

KODÉRY A DEKODÉRY

Kodér je kombinační síť hradel, které mění, konvertují (nebo kódují) nebinární vstup v binární. Například kodér pro převod osmičkové soustavy na dvojkovou má osm vstupů (jeden pro každou osmičkovou číslici). Logická jednička na jednom ze vstupů produkuje dvojkový ekvivalent na výstupu.

Kodér může provádět také další konverzní operace. Klávesnice kodéru například mění pozice jednotlivých kláves (klapek) na dvojková slova, která jsou jim přidělena. Příkladem je kód ASCII (American Standard Code for Information Interchange – Americký standardní kód pro přeměnu informací), v Evropě známý jako ISO 7, což je v podstatě zakódovaná klávesnice, u níž se po stisknutí klávesy vytváří sedmibitové slovo, např. zmáčkneme-li klávesu %, získáme na výstupu 0100101.

Dekodér je kombinační obvod, který mění dvojková čísla přiváděná na jeho vstupy na logické signály o úrovni log. 1 na jednom nebo několika jeho logických výstupech. V číslicové elektronice je totiž často nezbytné přeměnit dvojkové číslo v nějakou jinou formu, která je vyžadována k aktivování příslušných segmentů například v sedmisegmentovém displeji.

Dekodér se také používá k dekódování dvojkových (binárních) instrukcí mikroprocesoru, kde slouží k vytváření sekvenčních časových signálů pro složitější logické obvody a k přeměně dvojkových čísel v jejich osmičkové a šestnáctkové protějšky. Funkce kodéru a dekodéru je znázorněna na obr. 9.

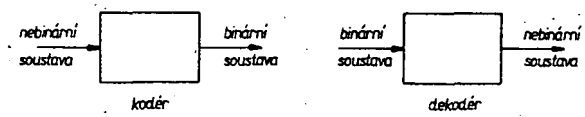
MULTIPLEXERY A DEMULTIPLEXERY

Multiplexer je číslicový logický obvod, ekvivalent mnohapolohového otočného přepínače. Typický multiplexer je kombinační logický obvod, který vybírá jednu nebo několik vstupních řádek a data na této řádce převádí k jednoduchému výstupu. Zvláštní sada adresovacích vstupů určuje, která vstupní řádka má být vybrána.

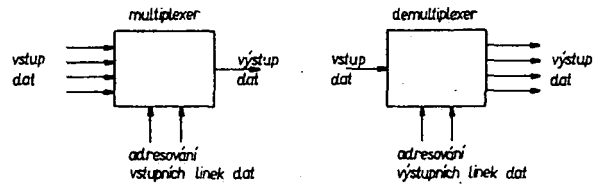
Typický multiplexer má osm datových vstupů, tři adresové vstupy a jednoduchý datový výstup. Když v adresovacích vstupech použijeme adresu 101, je vstup 5 spojen s výstupem.

Multiplexery se běžně používají k řízení zobrazovacích obvodů např. kapesních kalkulaček, aby se mohl co nejvíce omezit počet „nožiček" kalkulátorového čipu. Multiplexer aktivuje každou číslici nebo segment číslice na displeji mnohokrát za sekundu a tak „ošálí" naše oko, které vnímá údaj na displeji jako trvale svítící.

Demultiplexer převádí (přeměňuje) dvojkové údaje na vstupu na dvě nebo několik výstupních řádek. Stejně jako u multiplexeru řídí výstup adresový vstup. Demultiplexerů se používá ve spojení s multiplexery k tomu, aby přeměňovaly multiplexované údaje zpátky na jejich původní tvar. Demultiplexery mohou dokonce pracovat jako dekodéry, mají-li úroveň log. 1 na jednoduchém vstupu



Obr. 9. Kodér převádí nebinární soustavu na binární a dekodér pracuje opačně, převádí binární soustavu na nebinární



Obr. 10. Multiplexer je ekvivalentem několikapolohového přepínače a demultiplexer převádí multiplexovaná data do původního tvaru

a použijeme-li adresové vstupy jako vstupy pro data. Na obr. 10 je zjednodušeně znázorněna funkce multiplexerů a demultiplexerů.

SEKVENČNÍ LOGICKÉ OBVODY

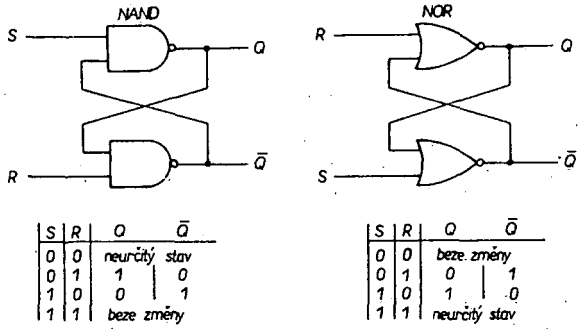
Na rozdíl od kombinačních logických obvodů mají sekvenční obvody paměť. Jejich výstup (výstupy) se chová nezávisle na tom, jaký byl stav na některém ze vstupů, a to stav před několika sekundami nebo dokonce před několika dny.

Nejjednodušším sekvenčním obvodem je klopný obvod („mezinárodně" nazývaný flip – flop). Mikroprocesor se čtecí a zároveň zaznamenávací pamětí (read/write memory) obsahuje desítky a možná i stovky klopných obvodů.

Klopných obvodů je několik druhů, všechny však jsou schopny uchovávat „v paměti" jednoduchý dvojkový bit. To umožňuje použít je v takových aplikacích, jako jsou čítače, děliče, paměťové registry aj. Uvedeme si čtyři základní druhy klopných obvodů.

Klopný obvod R-S

Nejjednodušší klopný obvod je vytvořen ze dvou hradel NAND nebo NOR s překříženými vstupy a výstupy (obr. 11). Tento základní obvod se nazývá nulovací – nastavovací (reset-set = R-S) klopný obvod nebo jednoduše západka. Na obr. 11 je též pravdivostní tabulka pro klopný obvod R-S jak z hradel NAND, tak NOR.



S	R	Q	$\bar{Q}$
0	0	neurčitý	stav
0	1	1	0
1	0	0	1
1	1	beze změny	

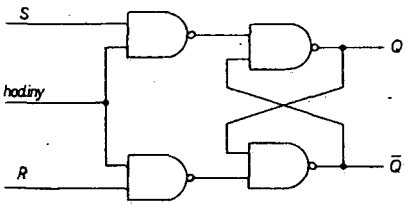
S	R	Q	$\bar{Q}$
0	0	beze změny	
0	1	0	1
1	0	1	0
1	1	neurčitý	stav

Obr. 11. Jednoduchý klopný obvod ze dvou hradel NAND nebo NOR se nazývá R-S. Pravdivostní tabulky ukazují jeho funkci

Povšimněme si toho, že dva výstupy z klopného obvodu R-S doplňují jeden druhý. Když je na Q úroveň log. 1, je klopný obvod nastaven („nahozen"). Je-li na Q úroveň log. 0, je klopný obvod vynulován.

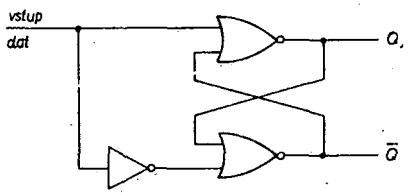
Hodinový klopný obvod R-S

Základní klopný obvod R-S je asynchronní; jeho odezva na změnu úrovně na vstupech je okamžitá. Činnost klopného obvodu R-S s dalšími logickými obvody lze synchronizovat např. hradlováním jeho vstupů tak, že obvod reaguje na stav vstupů pouze tehdy, jsou-li obvody aktivovány úrovní log. 1 z hodinového obvodu, což je sekvenční obvod, který produkuje střídavě úrovně log. 0 nebo log. 1. Na obr. 12 je klopný obvod R-S, řízený „hodinami" (clock), stručně se též nazývá hodinový klopný obvod R-S.



hodiny	S	R	Q	$\bar{Q}$
0	0	1	původní stav	
0	1	0	původní stav	
1	0	1	1	0
1	1	0	0	1

Obr. 12. Klopný obvod R-S řízený hodinovými impulsy je sekvenčním obvodem s pravdivostní tabulkou podle obrázku

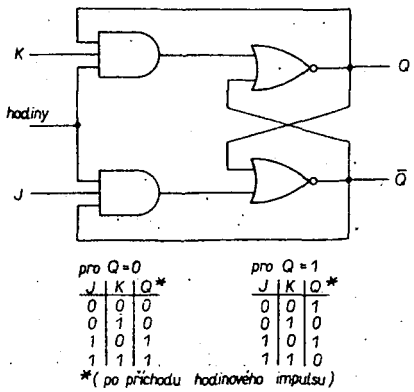


D	Q	$\bar{Q}$
0	0	1
1	1	0

Obr. 13. Klopný obvod typu D a jeho pravdivostní tabulka

Klopný obvod datový neboli D

Klopný obvod D je další modifikací hodinového klopného obvodu R-S. Jak je uvedeno na obr. 13, je k jednomu ze dvou vstupů tohoto klopného obvodu přidán invertor



pro Q = 0		
J	K	Q*
0	0	0
0	1	0
1	0	1
1	1	1

pro Q = 1		
J	K	Q*
0	0	1
0	1	0
1	0	1
1	1	0

*(po příchodu hodinového impulsu)

Obr. 14. Klopný obvod J-K s hodinovým vstupem a pravdivostní tabulkou před příchodem a po příchodu hodinového impulsu

a zbývající vstup a vstup invertoru jsou vzájemně spojeny. To zaručuje, že vstupy do sekce R-S tohoto klopného obvodu budou vždy doplňovat jeden druhý. A zajišťuje to také, že logický stav výstupu Q bude vždy odpovídat logickému stavu vstupu D.

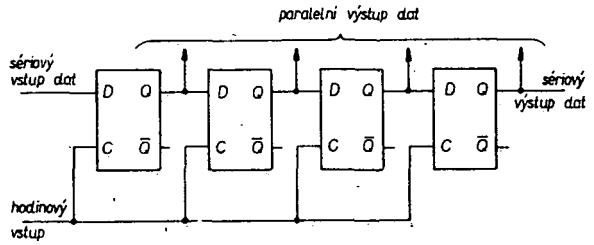
Klopný obvod J-K

Klopný obvod J-K je hodinový klopný obvod R-S s určitým „vylepšením", které dovoluje, aby úroveň log. 1 byla současně přivedena na oba vstupy. Logický obvod a jeho pravdivostní tabulka jsou na obr. 14. Klopný obvod J-K může snadno simulovat jakýkoli jiný druh klopných obvodů, takže se v sekvenčních logických obvodech používá velmi často.

Klopného obvodu J-K můžeme využít k tomu, abychom vytvořili klopný obvod T. Vstupy J a K jsou spojeny a nazývají se vstup T. Když je na T přivedena úroveň log. 1, mění klopný obvod svůj stav a sleduje hodinový impuls.

PAMĚŤOVÉ REGISTRY

Řada klopných obvodů D, nazývaná registr, může být použita k „uskladnění" nějakého dvojkového slova. Takový registr může být učiněn daleko užitečnějším tím, že k němu přidáme určitou kombinační logiku k současnému nulování všech klopných obvodů. Dále má registr vstup, který ovládá zápis dat do klopného obvodu. Takovéto registry pro uchování dat se někdy nazývají vyrovnávací registry. Využívá se jich v logických obvodech a v mikroprocesorech, aby si dočasně zapamatovaly nějaký bitový obrazec.



Obr. 15. Tento posuvný registr se skládá z klopných obvodů D a s příchodem každého hodinového impulsu posouvá data bit po bitu. Tento registr má jak paralelní, tak sériový výstup

POSUVNÉ REGISTRY

Mnohem univerzálnější než vyrovnávací registry jsou posuvné registry (obr. 15). Tento zvláštní registr přijímá synchronně s hodinovými impulsy vstupní data, zatímco dává současně k dispozici obsahy všech svých

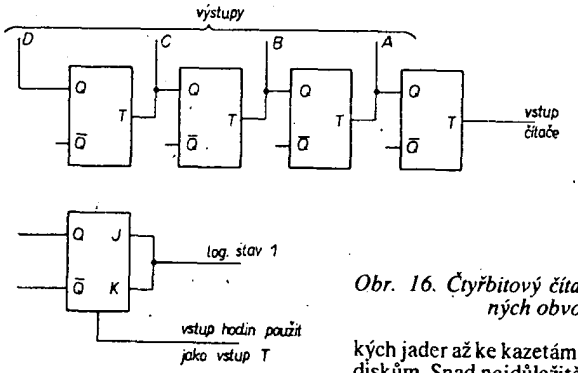
klopných obvodů (paralelní výstup). Datové bity v registru jsou přesouvány synchronně s hodinovými impulsy, aby se udělalo místo pro další vstupující bity.

Existují takové posuvné registry, které mohou přijímat a vydávat údaje sériově nebo paralelně, právě tak mohou posouvat data směrem zleva do prava nebo opačně. Činnost univerzálních posuvných registrů se řídí přivedením logických úrovní 0 nebo 1 na jejich řídicí vstupy. Mikroprocesory obsahují přinejmenším jeden posuvný registr, který je určen k nějaké manipulaci s daty, která se vyžaduje pro násobení, dělení dvojkových čísel, testování bitů apod.

ČÍTAČE

Jistě se ještě pamatujete na klopný obvod typu T. Výstup Q tohoto obvodu se překlápí z úrovně log. 0 na log. 1 při každém vstupujícím hodinovém impulsu. Jinými slovy: na výstupu Q je úroveň log. 1 pro polovinu vstupujících hodinových impulsů. To znamená, že tímto jednoduchým klopným obvodem lze dělit vstupující proud bitů dvěma. To však také znamená, že klopný obvod T na výstupu počítá od 0 do 1 ve dvojkové soustavě „pořád dokola".

Větší kapacity dvojkových čítačů (a děličů) může být dosaženo řadou klopných obvodů T, spojíme-li výstup Q jednoho klopného obvodu se vstupem dalšího klopného obvodu. Na obr. 16 je např. čtyřbitový čítač, vytvořený ze čtyř klopných obvodů T. Tento čítač bude počítat od 0000 do 1111 a pak se bude cyklus opakovat.



Obr. 16. Čtyřbitový čítač vytvořený z klopných obvodů T

Takových čítačů je velké množství. Modulo čítače určuje maximální počet impulsů, kterého bylo dosaženo před opakováním cyklu. Čítače modulo 10 jsou velmi populární, protože opakují cyklus po desátém vstupním impulsu a tím poskytují vhodný způsob k počítání v desítkové soustavě. Tyto čítače se často nazývají BCD (binary coded decimal – dvojkově kódovaná desítková soustava). Jejich čítací sekvence je:

$0000\ (0_{10})$
$0001\ (1_{10})$
$0010\ (2_{10})$
.
.
.
$1001\ (9_{10})$
$0000\ (0_{10})$
.
.
.

Čítače mohou mít různé řídicí vstupy. Typický čítač například může počítat nahoru nebo dolů. Může mít také řídicí vstupy pro vynulování obsahu na 0 nebo pro paralelní nastavení na jakoukoli požadovanou hodnotu a pro povolení čítat. A konečně protože čítače uchovávají akumulované počítání do příštího hodinového impulsu, mohou být považovány za paměťové registry.

PAMĚTI, LOGIKA ORIENTOVANÁ NA SBĚRNICE (BUS) A ORGANIZACE MIKROPROCESORU

Nyní si povíme něco o polovodičových pamětech a ukážeme si, jak třístavová logika dovoluje, aby logický obvod přenášel data k jednomu anebo k několika dalším obvodům přes pole vodičů, nazývané sběrnice neboli bus. Povíme si také o základech organizace mikroprocesoru.

Paměti

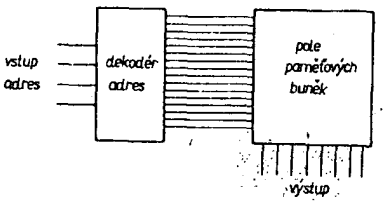
Samotný mikroprocesor je pouhou sbírkou logických obvodů na křemíkové destičce (čipu) a musí být proto vybaven podrobným seznamem instrukcí (nazývaných program), než může vykonávat nějakou užitečnou činnost. Tento program, společně se vstupními daty a výstupními daty z mikroprocesoru, je uchováván v paměti.

Paměť uchovává informace jako jednotlivé bity (nuly a jedničky) nebo jako bitové vzorce (slova). Jak jsme se již zmínili, může binární slovo indikovat nějakou číselnou hodnotu (data), nějakou adresu v paměti nebo počítačovou instrukci. Tak se stává paměť mimořádně mnohostranným zařízením a neodlučitelnou součástí mikroprocesoru.

Mikroprocesor můžeme používat ve spojení s jakýmikoli druhy pamětí, od magnetických jader až ke kazetám s pásky a k pružným diskům. Snad nejdůležitější jsou však paměti ROM a RAM. Polovodičové paměti ROM jsou paměti, z nichž je možno pouze číst (read only memory), RAM jsou paměti, do nichž je možno informace zapisovat, nebo je z nich číst (read/write memory) a obecně se dá říci, že data do nich uložená jsou dočasná, neboť mohou být snadno vymazána, popř. změněna.

Jak paměti ROM, tak paměti RAM jsou k dispozici jako integrované obvody s desítkami až stovkami jednotlivých paměťových buněk, které jsou včetně jejich spojů vytvořeny speciálními technologickými pochody.

Díky kombinačním logickým dekodérům na čipu je možný přístup ke všem paměťovým linkám dokonce i ve velmi rozsáhlé paměti pomocí relativně malého počtu adresovacích linek. Zjednodušeně je znázorněno to, jak dekodér toto adresování provádí, na obr. 17.



Obr. 17. Dekódování adresových linek v poli paměťových buněk pamětí ROM a RAM

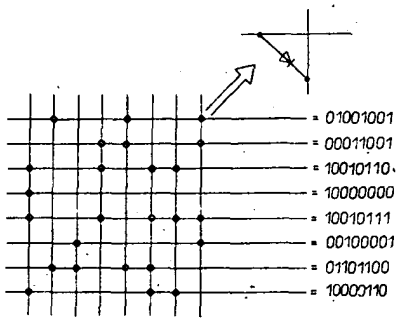
Jak můžete na obrázku vidět, pouhé přivedení příslušné adresy na adresové vstupy způsobí, že se požadovaný bit nebo slovo objeví na výstupní lince.

Paměťová data v RAM a ROM jsou jednotlivé bity nebo slova (obvykle čtyřbitové nibbly nebo osmibitové byty).

Dekodér zajišťuje přesně stanovenou a rychlou dobu přístupu k jakémukoli bitu nebo slovu v paměti. Tento jev se nazývá náhodný přístup. Paměti se sériovým vstupem, jako jsou magnetofonové pásky a posuvné registry s velkou kapacitou, jsou pomalejší, protože jejich obsah musí příslušné obvody prozkoumat bit po bitu, aby našly určenou adresu nebo určená data.

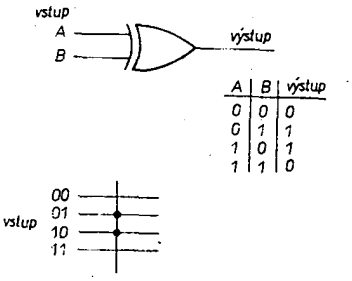
Paměti ROM

Typická paměť ROM je pole křižujících se vodičů (obr. 18). Dioda, která spojuje dva z těchto vodičů, představuje logickou jedničku, nepřítomnost diody v průsečíku logickou nulu. Informace je vepsána do paměti ROM již při výrobě a je trvalá, může být přečtena, avšak není ji možno změnit nebo doplnit.



Obr. 18. V paměti ROM je pole vodičů, které se kříží, avšak nejsou vodivě spojeny. Pouze v místech, kde je dioda, je vytvořen spoj, reprezentující stav log. 1

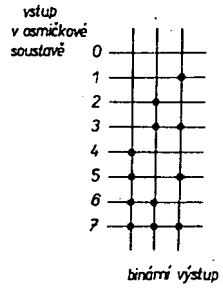
Paměť ROM může uchovávat binární data, adresy nebo instrukce. Může dokonce napodobovat logický obvod. Na obr. 19 je např. diodová paměť ROM, která je naprogramována pravdivostní tabulkou obvodu



Obr. 19. Paměťové pole paměti ROM, naprogramované podle pravdivostní tabulky obvodu EXCLUSIVE-OR

EXCLUSIVE – OR. Tento obvod v klasické formě vyžaduje přinejmenším čtyři logická hradla, z nichž každé obsahuje několik tranzistorů. Jak vidíte, je verze s pamětí ROM mnohem jednodušší a uplatní se u složitých kombinačních obvodů jako je dekodér instrukcí nebo znaků.

Obsah paměti ROM může být takový, aby paměť ROM napodobila skutečně jakýkoli



Obr. 20. Paměť ROM, naprogramovaná jako kodér osmičkové soustavy na soustavu dvojkovou

obvod kombinační logiky. To dokazuje např. obr. 20, kde je paměť ROM naprogramována jako kodér osmičkové soustavy na soustavu dvojkovou (obvod bývá obvykle navržen se sítí hradel OR. Navrhovat kodér z běžných logických obvodů je jak únavné, tak náročné na čas, kodér ROM si však může rychle navrhnout každý).

Vše, co je třeba k navržení kodéru z paměti ROM, je vhodná pravdivostní tabulka:

osmičkový vstup								binární výstup		
0	1	2	3	4	5	6	7	2^2	2^1	2^0
1	0	0	0	0	0	0	0	0	0	0
0	1	0	0	0	0	0	0	0	0	1
0	0	1	0	0	0	0	0	0	1	0
0	0	0	1	0	0	0	0	0	1	1
0	0	0	0	1	0	0	0	1	0	0
0	0	0	0	0	1	0	0	1	0	1
0	0	0	0	0	0	1	0	1	1	0
0	0	0	0	0	0	0	1	1	1	1

Zde si všimněme, že umístění diod v paměti ROM odpovídá polovině výstupů podle pravdivostní tabulky. ROM je nyní naprogramována jako kodér osmičkové soustavy na soustavu dvojkovou.

Používat paměť ROM je stejně snadné, jako navrhovat její obsah. Aktivujeme-li příslušnou vstupní řádku, objeví se určený bitový „vzorec" na výstupech sloupců.

Paměti ROM jsou k dispozici jako standardní obvody, naprogramované k plnění funkcí kodérů, dekodérů atd. Výrobci polovodičů mohou podle objednávky vyrobit paměť ROM na zakázku; taková paměť však by byla velmi drahá, pokud by nebylo objednáno velké množství kusů (několik tisíc). Jak to však udělat s pamětí ROM pro amatéry a kutily? Nejlepším řešením je programovatelná ROM čili PROM. Paměť PROM je v podstatě paměť ROM s diodami na všech jejích paměťových místech. Do paměti PROM je „vložena" požadovaná pravdivostní tabulka tím, že se přes vnitřní obvody přivedou na ty diody, které si přejeme odstranit, krátké elektrické impulsy. Tím se odstraní kovová vrstva (nazývaná též někdy tavná pojistka), která spojuje diodu s vodiči.

Paměti PROM stejně jako ROM nemohou být vymazány, když už byly jednou naprogramovány (i když samozřejmě lze odstranit další ze zbylých „tavných pojistek").

K dispozici jsou však i zvláštní vymazatelné paměti PROM. Jsou také naprogramovány elektricky, avšak vymazávají se ultrafialovým paprskem přes křemíkové okénko, které kryje křemíkovou destičku (čip). Ještě je nutno podotknout, že dnes již existují paměti, mazatelné elektricky.

Různé druhy pamětí ROM a PROM mohou uchovávat desítky až stovky bitů. Protože paměti ROM s kapacitou paměti od 2^6 (256) do 2^{16} (65 536) bitů jsou nejběžnější, označuje se často „obsah" pamětí ROM (i RAM) číslem s písmenem K (někdy velkým, někdy malým). Pak paměť 1K uchovává 1024 (2^{10}) bitů a paměť 4K 4096 (2^{12}) bitů.

Některé paměti uchovávají data jako jednotlivé bity. A tak 1 × 256 bitová ROM uchovává 256 bitů a 8 × 256 ROM uchovává 256 osmibitových bytů.

Paměti RAM

Paměti RAM, stejně jako paměti ROM se skládají z křižujících se vodičů na křemíkovém čipu. V místech křižování však nejsou v tomto případě diody, ale klopné obvody. Protože klopné obvody mohou být utvořeny tak, aby měnily svůj stav, znamená to, že data uchovaná v paměti RAM mohou být elektricky změněna nebo vymazána. To ovšem také znamená, že paměti RAM jsou mnohem složitější a tudíž také dražší, než paměti ROM.

Paměti ROM jsou klasifikovány jako stálé paměti, protože uchovávají informace bez přítomnosti elektrického proudu. Naproti tomu paměti RAM jsou nestálé: odpojíme-li napájecí napětí, jsou informace uchované v paměti RAM ztraceny, protože vnitřní klopné obvody se mohou po obnovení napájecího napětí nastavit do libovolného stavu.

Pamětí RAM se někdy využívá k uchování toho druhu informací, pro něž jsou vhodné paměti ROM; mnohem častěji se jich využívá k uchování dat a programů u mikroprocesorů, jako dočasných pamětí dat a pro jakékoli aplikace, které vyžadují paměti s rychle přeměnitelnou pravdivostní tabulkou.

Za paměť RAM, která může uchovávat čtyřbitové slovo, může být považován i jednoduchý čtyřbitový registr. Běžné paměti RAM mají však podstatně větší kapacitu paměti. Dnes jsou paměti RAM schopny uchovávat 16K (16 384) bitů a jsou již k dispozici paměti RAM 64K (65 536 bitů). Paměti RAM s tak velkou kapacitou mohou pracovat paralelně, aby mohly uchovávat několikanásobná bitová slova.

Další paměti

Nejdůležitějšími a nejpoužívatelnějšími pamětmi jsou polovodičové paměti RAM a ROM. Vyrábějí se však i další druhy pamětí a protože paměti hrají v práci mikroprocesorů důležitou roli, měli bychom o nich něco vědět.

Důležitou polovodičovou pamětí je paměť CCD (charge-coupled device – nábojově vázaný prvek). Tyto paměti uchovávají data jako elektrické náboje, které mohou být přemístěny z jedné buňky paměti do další. Přítomnost náboje reprezentuje log. 1, jeho nepřítomnost log. 0. Protože se k obsahu pamětí musí přistupovat sériově, jsou pomalejší než paměti RAM a ROM. Ale paměti CCD mohou uchovávat na křemíkovém čipu více dat, než paměti RAM a ROM podobné velikosti, protože neužívají dekodérů adres, které jsou třeba při náhodném vstupu.

Paměti z magnetických bublin mají velkou kapacitu a umožňují čtení a zápis a také stálé uchování dat. Bity se uchovávají jako přítomnost (1) nebo nepřítomnost (0) mikroskopických magnetických válečků, které se nazývají domény. Tyto válečky, díváme-li se na jejich konec mikroskopem, se podobají bublinám a mohou být rychle odstraněny.

Magnetopáskové paměti a pružné disky jsou běžně používány ke spolupráci s takovými mikroprocesorovými systémy, jako jsou třeba počítače. Existuje několik způsobů, jak uchovávat bity na magnetofonovém pásku. Jednou z možností je zakódovat logické úrovně jako dva různé tóny (tóny různého kmitočtu). Kazetové magnetofony jsou levné, snadno dostupné a ideální pro zavádění programů do pamětí RAM.

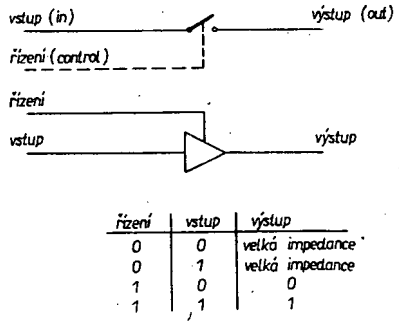
Pružný disk je něco jako gramofonová deska z pružné plastické hmoty, která má na povrchu nanesenu magnetickou látku, po dobnou té, která se používá k nahrávání u magnetofonového pásku. Bity jsou uchovány jako přítomnost nebo nepřítomnost zmagnetizovaných bodů asi na stovce nebo i více stopách umístěných na povrchu disku. Disk se otáčí velkou rychlostí a čtecí a záznamová hlava na pohyblivé dráze umožňuje přístup k jakékoli stopě záznamu. Pružné disky poskytují paměť s velikou kapacitou a s pozoruhodně krátkou vybavovací dobou, mnohem kratší, než u magnetofonového pásku. Systémy s pružnými disky jsou však drahé; stojí často víc než počítač, neboť mají složitý řadič a mechanika je náročná na přesnost výroby.

TŘÍSTAVOVÁ LOGIKA

Zatím jsme si stručně probrali základní logické obvody, kombinační a sekvenční logické obvody a paměti. Můžeme tedy pomalu začít používat tyto obvody jako elektronické stavební bloky ke konstrukci mikroprocesoru. Zbývá nám již jen zmínit se o novém druhu logického obvodu, nazývaného třístavový výstup.

Jak je možno vidět na obr. 19 a 20, jsou výstupy ze všech logických obvodů, které jsme dosud probrali, různými kombinacemi dvou stavů, log. 0 a 1 – proto se logika, která pracuje s těmito obvody, nazývá dvoustavová (dvouhodnotová). Třetí stav, nazývaný stav s velkou impedancí, je užíván ve třístavové logice. V tomto třetím stavu je výstup elektronicky odpojen od vstupů. Je to jako by byl mezi vstupem a výstupem obvodu zařazen spínač – je-li vypnut, to znamená není-li na vstupu stav s velkou impedancí, objevují se na výstupu logické stavy log. 0 a log. 1.

Je-li na řídicím vstupu úroveň log. 1, přenáší třístavové hradlo podle obr. 21 určitý logický stav (log. 0 nebo log. 1) ze svého vstupu na výstup. Je-li na řídicím vstupu úroveň log. 0, pak je na výstupu stav „velká impedance" a výstup se chová tak, jako kdyby byl odpojen.



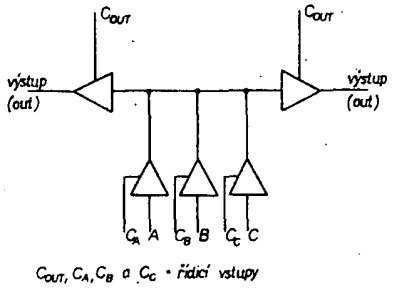
řízení	vstup	výstup
0	0	velká impedance
0	1	velká impedance
1	0	0
1	1	1

Obr. 21. Jednoduché třístavové hradlo může být reprezentováno spínačem. Řídicí vstup ovlivňuje činnost hradla, jak ukazuje pravdivostní tabulka

Všechny základní typy hradel jsou vyráběny i ve třístavové verzi a stejně tak se vyrábí s třístavovými výstupy mnoho druhů složitějších obvodů, jako jsou klopné obvody, čítače, registry a kombinační sítě. Třístavová logika umožňuje spojit výstupy dvou nebo několika logických obvodů s vodičem, který se nazývá sběrnice (bus). U dvoustavové logiky není možno spojit výstupy dvou nebo více hradel, protože na některých výstupech může být

úroveň log. 0 a na jiných úroveň log. 1. U třístavové logiky je možno připojit ke stejné sběrnici výstupy mnoha hradel, logickou informaci však přivádí na sběrnici v každém okamžiku pouze jeden výstup a ostatní musí být ve stavu „velká impedance".

Několik třístavových vyrovnávacích obvodů podle obr. 21 bývá připojeno k běžné sběrnici podle obr. 22.



C_{OUT}, C_A, C_B a C_C - řídicí vstupy

Obr. 22. Třístavové obvody připojené ke sběrnici. Data na této sběrnici mohou „putovat" oběma směry (zleva doprava a naopak), proto se tato sběrnice nazývá obousměrná

Podívejme se na chvíli na tento obrázek. Všimněme si, že použití řídicích vstupů (C) dovoluje, aby přenos dat byl řízen z kteréhokoli ze tří vstupů k jednomu nebo oběma výstupům. Tato činnost je podobná činnosti multiplexeru a tříhodnotová logika bývá někdy používána k napodobování multiplexeru. Ještě důležitější je to, že toto uspořádání dovoluje, aby data „cestovala" podél sběrnice v obou směrech. Proto se tříhodnotová sběrnice někdy nazývá také obousměrná.

PŘENOS DAT – „Z REGISTRU DO REGISTRU"

Typický mikroprocesor obsahuje několik registrů pro uchování dat. Třístavová logika poskytuje účinný způsob, jak přenášet data jednoho z těchto registrů do druhého. Na obr. 23 jsou tři čtyřbitové registry spojené běžnou čtyřvodičovou sběrnicí. Výstup každého registru je napojen na sběrnici prostřednictvím čtyřbitového vyrovnávacího obvodu. Proto jak vstupní, tak výstupní linky registrů mohou být napojeny na tutéž sběrnici.

Každý registr na obr. 23 má tři řídicí vstupy: čtecí (snímací), záznamový a hodinový.

Úroveň log. 0 na řídicím vstupu čtení způsobí, že výstup registru je ve stavu „velká impedance" a izoluje tak data uchovaná tomto registru od sběrnice a tím také od dalších registrů. Naproti tomu úroveň log. 1 na řídicím vstupu pro čtení způsobí, že tříhodnotové vyrovnávací hradlo přenese data z tohoto registru na sběrnici. Povšimněme si, že pouze jeden registr může dodávat v jednom časovém okamžiku svůj obsah na sběrnici, jinak by mohl na sběrnici „vznikout zmatek".

Data, která jsou na sběrnici, mohou být zapsána (přenesena) do jednoho nebo několika registrů tím, že na příslušný záznamový vstup je přivedena úroveň log. 1. Když přijde další hodinový impuls, zaznamenají se data do vybraných registrů.

Pokusme se přenést data z registru A do registru C – jak je to znázorněno na obr. 23: Nejprve přivedeme na čtecí vstup A registru A/R úroveň log. 1. Potom přivedeme úroveň log. 1 na záznamový vstup registru C. Když přijde hodinový impuls, bude obsah registru A „přenesen" do registru C. Registr A si může nadále uchovávat svá data, ale registr B bude vyprázdněn.

Tohoto jednoduchého postupu lze využívat k přenosu obsahů registru A, B nebo C do jednoho nebo obou ze zbývajících registrů. Na kterých řídicích vstupech musí být úroveň log. 1, aby byl přenesen obsah registru C do registru A a B?

POJEM ŘÍZENÍ

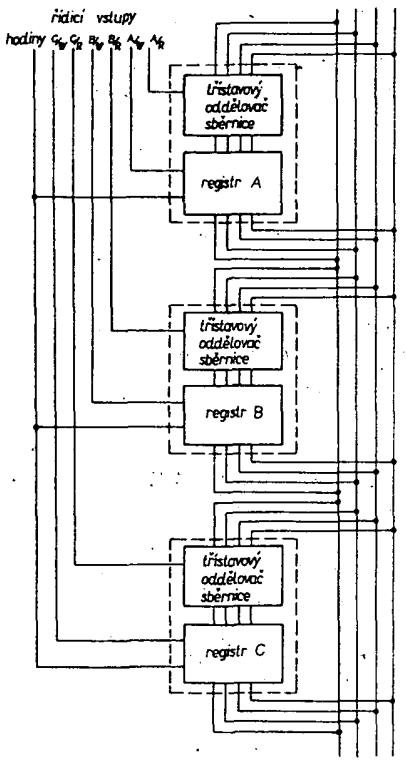
Dospěli jsme tedy postupně tak daleko, že bychom se mohli podívat na to, jak je sestaven dohromady mikroprocesor. Zamyslete se však ještě nad řídicími vstupy tří registrů, znázorněných na obr. 23.

Existuje 9 způsobů, jak přenášet data mezi těmito registry: A do B, A do C, B do A, B do C, C do A, C do B, A do B a C, B do A a C, C do A a B. Nejjednodušším způsobem, jak kategorizovat volby těchto přenosů dat, je sestavit jejich výčet s uvedením obrazců bitů, které jsou vyžadovány na každém z řídicích vstupů, jak je ukázáno v následující tabulce:

operace	řídicí vstupy					
	A/R	A/W	B/R	B/W	C/R	C/W
A B	1	0	0	1	0	0
A C	1	0	0	0	0	1
B A	0	1	1	0	0	0
B C	0	0	1	0	0	1
C A	0	1	0	0	1	0
C B	0	0	0	1	1	0
A B & C	1	0	0	1	0	1
B A & C	0	1	1	0	0	1
C A & B	0	1	0	1	1	0

Nyní je každá z možností přenosu určena svým vlastním binárním řídicím slovem. V mikroprocesorech se tato řídicí slova, která přenášejí data mezi registry a provádějí mnoho jiných dalších operací, nazývají mikroinstrukce.

Často je nutné uskutečnit mezi registry několik přenosů, jeden po druhém. Napří-



Obr. 23. Tři čtyřbitové registry, spojené se čtyřbitovou třístavovou sběrnicí. Tři vstupy jsou: čtení (read), zápis (write) a hodinový vstup (clock)

klad jednou z možných sekvencí, která je použita na obvodu podle obr. 23, je přenos dat z registru A do B, B do C a C do A. Z tabulky, která je výše uvedena, jsou mikroinstrukce, kterých je k těmto operacím třeba, tyto:

1'00100
001001
010010

V mikroprocesorech se sekvence mikroinstrukcí, které provádějí specifickou řadu operací (jako jsou třeba ty, které jsme si uvedli), nazývají mikroprogram (microroutine). Mikroprocesory mají zvláštní sekci (řadič), která (kromě jiného) generuje mikroprogramy, které jsou potřeba k přenosu dat uvnitř mikroprocesoru.

MIKROPROCESOR

Funkce konvenčního číslicového integrovaného obvodu nemůže být významně změněna, aniž by nebylo z valné části změněno jeho zapojení. Mikroprocesor je naproti tomu docela jiný. Může být zkonstruován tak, aby vykonával mnoho různých funkcí jenom díky tomu, že se změní sekvence binárních slov nebo instrukcí uchovávaných v jednom nebo několika paměťových čipech, které jsou vně mikroprocesoru.

Skutečnost, že mikroprocesor lze naprogramovat, ho velmi přibližuje základní jednotce samočinného počítače. Přidáme-li k mikroprocesoru vnější paměťový obvod k uchování instrucí a dat, stává se mikroprocesor mikropočítačem. Některé poslední modely mikroprocesorů již obsahují na jednom čipu paměť pro instituce a data. Tyto prvky se nazývají jednočipové mikropočítače.

I když lze mikroprocesor používat jako část počítače, jsou ještě další nesčetné, avšak stejně důležité aplikace, od zařízení řídicích signalizací a např. elektronických vah, až po zkušební a měřicí přístroje, které se samy nastavují a cejchují, a „chytré" kapesní kalkulačky. V mnohých těchto aplikacích je program mikroprocesoru uchován trvale v pamětech ROM. Některé z těchto typů pamětí, obsahující různé programy, mohou být použity s tímtéž mikroprocesorem ve zcela odlišných aplikacích.

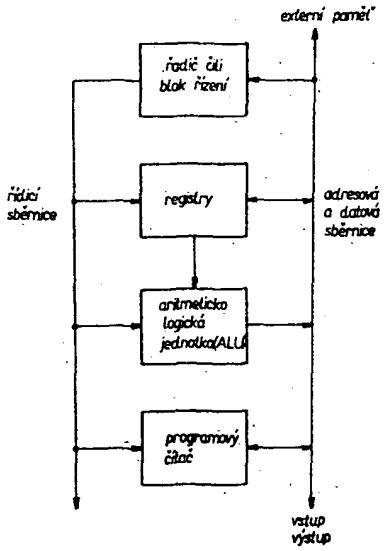
ORGANIZACE MIKROPROCESORU

„Minimální" mikroprocesor obsahuje řídicí sekci, programový čítač, který sleduje posloupnost prováděných instrukcí a data uložená ve vnějších pamětech, několik registrů dat a instrukcí a konečně aritmetickologickou jednotku (ALU). Jeden způsob, jak mohou být tyto obvody organizovány s ohledem na jejich vzájemnou činnost a na činnost řídicí sběrnice, adresové i datové sběrnice, aby mohl být sestaven mimořádně jednoduchý mikroprocesor, je na obr. 24.

Na obr. 24 není nic neobvyklého nebo složitého. Nejdůležitější je způsob, jakým jsou propojeny tyto části s oběma sběrnicemi. Podívejme se na některé z operací, které realizují jednotlivé sekce našeho procesoru.

Řídicí sekce

Řídicí sekce je nervovým centrem mikroprocesoru. Typický mikroprocesor může vykonávat 50 nebo více nejrůznějších operací, a to téměř v jakékoli kombinaci nebo v jakémkoli pořadí. (Později se na některé významnější instrukce podíváme podrobněji). Úlohou řídicí sekce je dopravovat instrukce – jednu za určitý časový úsek – z programové



Obr. 24. Organizace základní jednotky mikroprocesoru. Řídicí sekce je nervové centrum mikroprocesoru. Programový čítač sleduje plněné instrukce krok po kroku. Registry slouží k ukládání informací a ALU k vykonávání aritmetických nebo logických operací

paměti ROM nebo RAM, spojené s adresovou a datovou sběrnicí mikroprocesoru, dekódovat je a pak je vykonávat pomocí mikroinstrukcí; a potom vyvolat další instrukci.

Čítač instrukcí

Čítač instrukcí sleduje adresy prováděných instrukcí a obsahuje vlastně adresu, z níž bude vzata následující instrukce. Je to jednoduchý čítač, jehož výstupy jsou používány jako adresové vstupy k vnější paměti obsahující program a data, která se zpracovávají mikroprocesorem.

Čítač instrukcí a adresová a datová sběrnice určují, kolik slov z vnější paměti může být přijato mikroprocesorem. Proto čtyřbitový čítač instrukcí může mít přístup k paměti o 16 slovech (2^4), osmibitový čítač instrukcí může mít přístup k paměti o 256 slovech (2^8), šestnáctibitový čítač může mít přístup k paměti o 65 536 slovech (2^{16}).

Běžně postupuje čítač instrukcí programem vždy o jeden krok za určitou dobu ve vzestupném číselném pořádku, určité instrukce však mohou „naplnit" čítač instrukcí novým slovem, které je potom použito jako další vnější paměťová adresa. To dovoluje mikroprocesoru, aby „odbočoval" nebo „skákal" na různé části programu, nebo aby „prokličkoval" určitými částmi programu více než jednou.

Větvení a kličkování může být nepodmíněné nebo podmíněné. V druhém případě bude čítač instrukcí dostávat novou adresu pouze tehdy, budou-li splněny určité podmínky – bude-li například výsledek předcházejícího výpočtu negativní apod.

Registry

Mikroprocesor má několik registrů pro dočasné uchování dat, adres a instrukcí. Adresový registr uchovává adresy, z nichž mají být čtena data nebo instrukce, do té doby, dokud řídicí sekce nepožaduje novou adresu. Registr instrukcí uchovává instrukce přečtené z vnější paměti, dokud není instrukce vykonána a dokud není přečtena nová instrukce. Různé datové registry uchovávají slova, která čekají na další zpracování a působí jako výstupní vyrovnávací paměti (buffers).

Registr sčítačky uchovává mezivýsledky a konečné výsledky operací aritmetickologické jednotky a říkáme mu střadač (nebo také někdy akumulátor). Může mít schopnost přičítat (přidávat + 1) nebo odečítat jednotku od nějakého slova, právě tak jako posunovat slovo vlevo nebo vpravo. Často musí data vstupující a opouštějící mikroprocesor projít střadačem. A tak se dá říci, že střadač je v mikroprocesoru nejdůležitějším registrem.

Aritmetickologická jednotka (ALU)

Aritmetickologická jednotka může vykonávat aritmetické nebo logické operace na jednom nebo dvou datových slovech. Střadač je s aritmetickologickou jednotkou v úzkém spojení. Zcela typické je, že jedno ze slov, které má být zpracováno v aritmetickologické jednotce, dodává střadač. Výsledek je pak z výstupu aritmetickologické jednotky dodán zpátky do střadače přes adresovou a datovou sběrnici.

PROGRAMOVÁNÍ MIKROPROCESORU

Dosud jsme kladli důraz na hardwarový aspekt mikroprocesorů. Hardware je důležitý, ale bez software, bez programů, které „říkají" mikroprocesoru, co má dělat, není mikroprocesor k ničemu. Mohlo by se říci, že software je pro mikropočítač asi tolik, jako recept pro kuchaře.

Mikroprocesor má ve svém souboru desítky instrukcí a je úkolem programátora zkombinovat některé z nich nebo všechny tak, aby mikroprocesor mohl vykonávat daný úkol.

Běžnou instrukcí mikroprocesoru je „naplnit" střadač, nebo jednoduše LDA (Load Accumulator). Jak je uvedeno v závorce, je LDA vlastně zkratkou, vytvořenou z instrukce. Tyto zkratky se nazývají mnemotechnické. Instrukce LDA vloží do sumátoru datové slovo, které ji následuje v programu.

Další běžnou instrukcí pro mikroprocesor je JMP (přeskoč bezpodmínečně k určené adrese); dále JZ (přeskoč pouze tehdy, je-li na určitém klopném obvodu 0); dále je to JP (přeskoč pouze tehdy, je-li výsledek operace pozitivní); CLA (vynuluj střadač); ADD (přidej k obsahu střadače obsah datového registru a výsledek uchovej ve střadači); MOV (přenes obsah jednoho určitého registru do druhého); RAL nebo RAR (rotuj bity ve střadači zleva nebo zprava); HLT (zastav mikroprocesor) a jiné.

Tyto instrukce jsou pouze „zástupci" instrukcí, které má skutečný mikroprocesor k dispozici, dobře však ilustrují „počítačové možnosti" mikroprocesoru.

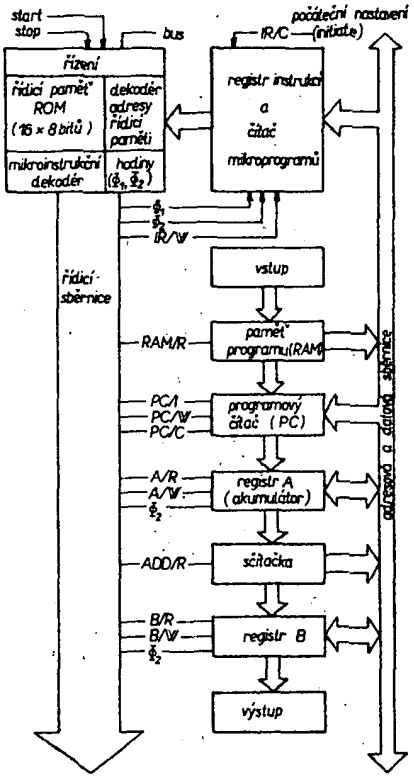
MIMOŘÁDNĚ JEDNODUCHÝ VYUČOVACÍ MIKROPROCESOR

Nyní se podíváme na PIP-2, což je velmi jednoduchý, čtyřbitový vyučovací mikroprocesor. I když PIP-2 je mnohem jednodušší než mikroprocesory 8080, Z80, 6502, 2650 a další mikroprocesory v pravém slova smyslu, ilustruje některé důležité pracovní rysy mikroprocesorů.

Programovatelný instrukční procesor PIP-2 má mnoho prvků dokonale propracovaného mikroprocesoru. Tak např. obsahuje vestavěnou programovatelnou paměť – to ho kvalifikuje do třídy mikropočítačů. Protože obsahuje i mikroprogramovatelnou řídicí paměť ROM, znamená to, že jeho soubor programů může být revidován nebo vyměněn a nahrazen jinými, zcela novými instrukcemi (o tom si povíme později).

Organizace PIP-2

Blokový diagram hlavních částí PIP-2 je na obr. 25. Jak je z obrázku zřejmé, je PIP-2 mikroprocesor organizovaný sběrnicově.



Obr. 25. Blokové schéma organizace mikroprocesoru PIP-2

Všechny jeho části jsou spojeny s čtyřbitovou obousměrnou sběrnicí, která dovoluje, aby byly přiváděny jak adresy paměti, tak data. Vzpomínáte si jistě, že jsme se již zmínili o tom, že při obousměrné sběrnici může číst data v jednom časovém okamžiku pouze jedna funkční část připojená na tuto sběrnici. PIP-2 splňuje toto pracovní omezení tím, že všechny jeho části, které jsou určeny ke snímání dat ze sběrnice, mají tříhodnotové výstupy. Tím jsou výstupy těchto částí PIP-2 izolovány od sběrnice do té doby, dokud nejsou aktivovány příslušným signálem z řídicí sekce.

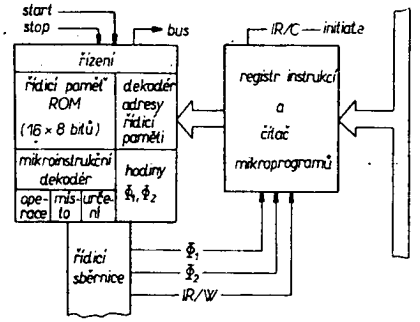
Podívejme se dále na jednotlivé části vyučovacího mikroprocesoru PIP-2.

Vstup

Jako vstupy slouží čtyři klopné obvody, působící jako spínače. Vstupy jsou na čelním panelu uspořádány tak, aby PIP-2 mohl sloužit jako mikropočítač (obr. 26).

Tím, že zavedeme do PIP-2 napájecí napětí, vynulují se registry A a B, čítač instrukcí a programová paměť RAM. To umožňuje, aby byl program do programové paměti zaveden pouhým zapnutím binární instrukce nebo datového slova a stlačením tlačítka pro vstup (LOAD).

Do programové paměti můžeme vložit až šestnáct čtyřbitových instrukcí a datových slov. Poté, co vložíme program, je čítač instrukcí (programový čítač) vynulován na 0000 tím, že zmáčkneme tlačítko pro začátek (INITIATE). Tak se vrátí čítač k první adrese v programové paměti a je připraven pro „sjetí" programu.



Obr. 28. Organizace řídicí sekce mikropočítače PIP-2

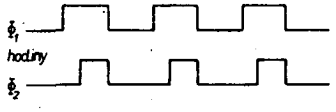
Všechno, co jsme si uvedli, se může zdát jako mimořádně složité, ale ve skutečnosti je to jednoduché, protože programová instrukce je pouhý bitový obrazec, který může být interpretován touto řídicí sekcí tak, aby vykonal určitý úkol. Pokud to ještě zjednodušíme: řídicí sekce není o nic komplikovanější (alespoň v zásadě) než obvod dekodéru, který rozsvěcuje příslušné segmenty sedmisegmentového displeje jako odpověď na binárně zakódovaný desítkový nibble.

Srdcem řídicí sekce některých mikroprocesorů je složitá kombinační síť (gates), která dekóduje programové instrukce a aktivuje příslušný řídicí vstup různých sekcí procesoru. Složitější mikroprocesory mají zvláštní paměť ROM, která obsahuje řady mikroinstrukcí nezbytných k provádění každé programové instrukce. Tyto tzv. mikroprogramovatelné mikroprocesory jsou mnohostrannější než běžné typy, protože jejich soubory instrukcí mohou být ve značné míře revidovány jednoduchou modifikací mikroinstrukcí uchovávaných v ROM. PIP-2 je mikroprogramovatelný a blokový diagram na obr. 28 ilustruje všeobecnou organizaci jeho řídicí sekce. V předešlém textu již bylo vysvětleno, jak řízení spolupracuje se zbývající částí PIP-2.

Na obr. 29 je podrobně znázorněn systém řízení PIP-2, včetně organizace mikroprogramovatelné ROM, obsahující mikroinstrukce, dekodér mikroinstrukcí a hodiny. Každé části této řídicí sekce si dále všimneme poněkud podrobněji.

Hodiny

Generátor hodinových impulsů, „hodiny", je relativně jednoduchou, ale životně důležitou částí řízení, protože poskytuje impulsy synchronizující jednotlivé cykly programu. O výstupu hodin se říká, že je dvoufázový, dodává-li dvě skupiny impulsů se stejným kmitočtem, ale s různou fází (výstupy Φ_1 a Φ_2). Časovací diagram pro tyto dva hodinové signály je na obr. 30.



Obr. 30. Časový diagram „dvoufázových hodin" u PIP-2

Hodiny mají dva řídicí vstupy. Úroveň log. 0, přivedená na řídicí vstup C/S, po stisknutí tlačítka START, uvádí hodiny do chodu. Úroveň log. 0, přivedená na řídicí vstup C/D po stisknutí tlačítka STOP, popř. signál z dekodéru mikroinstrukcí (aktivovaného mikroinstrukcí HLT v programu), hodiny zastavuje (generátor hodinových impulsů nepracuje).

Registr instrukcí a čítač mikroprogramů

Je to čtyřbitový čítač, který v našem případě pracuje i jako čtyřbitový registr. Na jeho vstupy se přivádí operační kód z programové paměti, což jsou ve skutečnosti adresy řídicí paměti ROM, které se přes výstupy vkládají do dekodéru adres řídicí paměti ROM.

Hodinové signály Φ_1 určují přírůstek registru instrukcí a působí, že postupuje řadou adres v řídicí paměti ROM, podobně jako programový čítač postupuje adresami v programové paměti, když vykonává program. Proto také můžeme registr instrukcí nazývat čítačem mikroprogramů. Tento registr má dvojici řídicích vstupů. Je-li na řídicím vstupu IR/W úroveň log. 0, zaznamenává se při příchodu hodinového impulsu Φ_2 jeho stav na adresovou a datovou sběrnici a do registru instrukcí. Je-li na druhém řídicím vstupu IR/C úroveň 0, pak je registr instrukcí vynulován na stav 0000.

Dekodér adres řízení paměti ROM

Jedná se o jednoduchý dekodér 1–ze–16, který aktivuje příslušné adresy v řídicí paměti ROM jako odpověď na přicházející data z registru instrukcí. Když je na vstupy dekodéru přiveden nibble 0000, pak je vybrána první adresa v řídicí paměti ROM.

Řídicí paměť ROM

Je to 128bitová paměť ROM, organizovaná jako šestnáct 8bitových bytů. Každý byte má přidělenu jednu adresu (od 0000 do 1111) a obsahuje nějakou jednoduchou mikroinstrukci. Jak je ukázáno na obr. 29, je řídicí paměť ROM vybavena mikroprogramy (řadami mikroinstrukcí) pro šest oddělených programových instrukcí. Jak brzy uvidíme, mohou být tyto mikroprogramy snadno změněny pouhou výměnou paměti ROM nebo jejím případným přeprogramováním.

Dekodér mikroinstrukcí

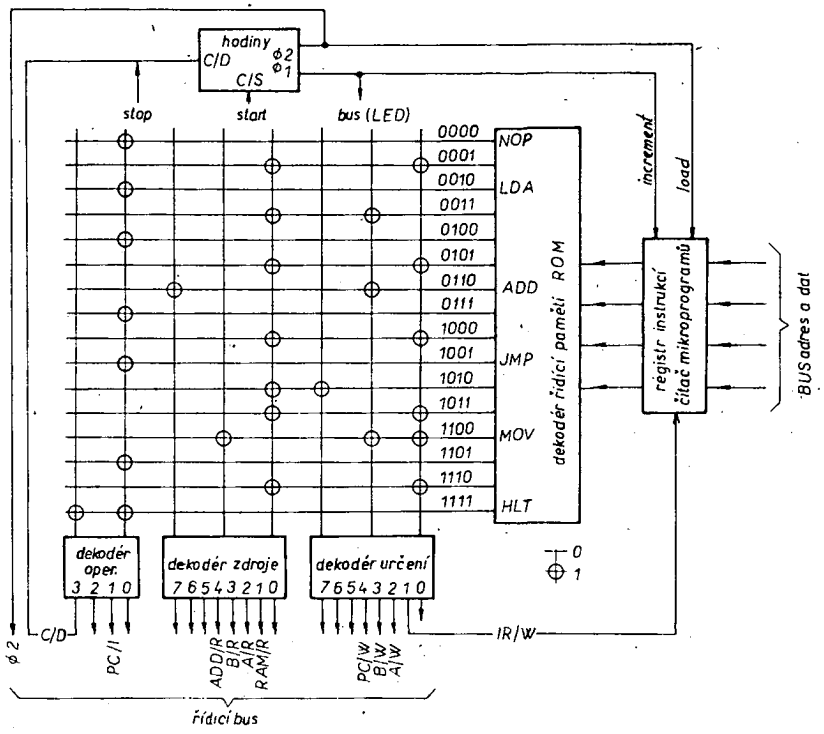
Řízení má dva dekodéry 1–z–8 (zdroj a určení) a jednoduchý dekodér 1–ze–4 (činnost). Vybraný výstup každého dekodéru je ve stavu log. 0, zatímco zbývající výstupy zůstávají ve stavu log. 1.

Tyto dekodéry přeměňují mikroinstrukce zakódované ve vybrané adrese paměti ROM do mikroinstrukcí, které jsou nezbytné k vykonávání určitých operací. Jak můžete vidět na obr. 29, je řídicí paměť ROM rozdělena do šestnácti osmibitových bytů. První dva bity z každého bytu jsou vloženy do dekodéru operací. Další tři bity jdou do dekodéru zdroje a poslední tři bity jdou do dekodéru určení.

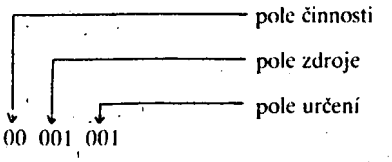
Výstupy z těchto tří dekodérů a oba hodinové signály tvoří řídicí sběrnici programovatelného instrukčního počítače PIP-2. Výstupy z dekodéru zdroje jsou přivedeny na řídicí vstupy snímání (R – read) různých sekcí PIP-2. Výstupy z dekodéru určení jdou do řídicích vstupů záznamu (W – write) různých sekcí. Výstupy dekodéru činností jdou do řídicích vstupů zvláštní činnosti jako je blokování hodin (C/D) a čítače přírůstku paměti (PC/I).

Všimněte si, že několika výstupů dekodérů zdroje a určení dvou výstupů z dekodéru činností se vůbec nevyužívá. To znamená, že u PIP-2 lze přídavné obvody (dejme tomu registr C nebo ALU) připojit ke sběrnici adresových dat. Těchto linek můžeme využít také k řízení vnějších přídavných zařízení. V obou případech by ovšem musely být přidány k řídicí paměti ROM nové mikroinstrukce, aby se aktivovaly nové obvody.

Věnujte také pozornost tomu, jak obrazce bitů uchovávaných v paměti ROM aktivují dekodéry. Tak například adresa 0001 obsahuje mikroinstrukci 00001001. Rozdělme tento byte do tříbitových polí, aplikovaných na dekodér, a uvidíme, co se stane:



Obr. 29. Vnitřní uspořádání řídicí sekce PIP-2 ukazuje, jak tři dekodéry spolupracují s ROM. Paměť ROM je znázorněna jako síť protínajících se vodičů, jejichž spojení odpovídá log. 1 a křížování log. 0



Pole činnosti (00) nedělá nic, protože aktivuje nespojený výstup 0 dekodéru činnosti.

Pole zdroje (001) aktivuje výstup 1 dekodéru zdroje. Znamená to stav log. 0 na řídicím vstupu RAM/R.

Pole určení (001) aktivuje výstup 1 dekodéru určení, což znamená, že na řídicím vstupu IR/W bude stav log. 0.

A výsledek? Výstup z programové paměti (RAM) a vstup do registru instrukcí (IR) jsou současně připojeny na adresovou a datovou sběrnici. Při příchodu hodinového impulsu Φ_2 se naplní registr instrukcí vybraným instrukčním operačním kódem.

Nyní, když už víme něco o každé ze sekcí PIP-2 a některé podstatné informace o tom, jak se jednotlivé mikroinstrukce vykonávají, podívejme se, jak řídicí sekce vyvolávají a vykonávají některou z instrukcí programové paměti.

Vyvolání a vykonání instrukcí

Když pochopíte, jak řídicí sekce vyvolává instrukci z programové paměti a jak ji vykonává, hodně vám to pomůže v pochopení toho, jak skutečný mikroprocesor pracuje. Pro lepší pochopení je vhodné opsat si soubor mikroinstrukcí PIP-2.

Předpokládejme, že první instrukcí v programové paměti (adresa 0000) je instrukce LDA. Je to paměťová referenční instrukce, která je následována čtyřbitovým datovým nibblem v další adrese programové paměti. Když je instrukce vykonána, naplní LDA registr A datovým nibblem adresy 0001 programové paměti (RAM).

Poté, co je program, obsahující instrukci LDA, vložen do programové paměti, je tlačítkem INITIATE vrácen programový čítač k adrese 0000 programové paměti. Registr instrukcí, jak již bylo vysvětleno, se vynuluje tlačítkem INITIATE také.

Dvě mikroinstrukce, které obsahuje instrukce NOP, zaujímají první dva byty v řídicí paměti ROM. Když se stiskne tlačítko START, posune první hodinový impuls čítač mikroprogramů ke druhé mikroinstrukci NOP, což je adresa v řídicí paměti ROM 0001.

Z obr. 29 je vidět, že tato mikroinstrukce je 00001001. Tato mikroinstrukce aktivuje řídicí signály RAM/R a IR/W, o nichž jsme dříve diskutovali. Když přijde hodinový impuls Φ_2, okopíruje registr instrukcí operační kód instrukce na adrese 0000 programové paměti. Operační kód instrukce LDA je 0001, takže v tomto případě registr instrukcí nemění stav. (Co by se stalo, kdyby operační kód této instrukce byl 1011 nebo 0101?)

Až sem jsme popisovali případ, kdy byly všechny řídicí operace naprogramovány a úplně automatizovány se specifickým cílem vyvolat první instrukci z programové paměti. Co se stane dál?

Připomeňme si, co to je operační kód pro každou instrukci. Je to dvojkové číslo, které je o 0001 menší než počáteční adresa mikroprogramu v řídicí paměti ROM, která vykonává tuto instrukci. Když přijde další hodinový impuls Φ_1, posune se registr instrukcí k první mikroinstrukci v programu LDA a tím se začnou plnit dílčí úkony mikroinstrukce. Sledujme dílčí kroky při plnění mikroprogramu LDA.

První mikroinstrukce LDA (viz obr. 29) je 01000000. Pouze řídicí signál PC/I je aktivován; čítač programů je posunut k další adrese v programové paměti, která obsahuje datový nibble, který má být vložen do registru A. Hodinový impuls Φ_2 je v podstatě signál pro činnost „nedělej nic", protože na adresové a datové sběrnici nejsou umístěna žádná data. Třetí hodinový impuls Φ_1 posouvá registr instrukcí ke druhé mikroinstrukci v mikroprogramu (00001010), který vyvolá stav log. 0 na řídicích signálech RAM/R a A/W. Když přijde hodinový impuls Φ_2, je obsah datového nibblu zapsán do registru A.

Nyní, když byl do registru A vložen určitý datový nibble, byla provedena nejdůležitější část instrukce LDA. Zbývající dvě mikroinstrukce vyvolávají další krok z programové paměti.

Čtvrtý hodinový impuls Φ_1 posouvá registr instrukcí k mikroinstrukci 01000000 LDA. Ta posouvá programový čítač k další adrese v programové paměti (0011). Následující hodinový impuls Φ_2 je dalším impulsem pro operaci „nedělej nic". Pátý hodinový impuls Φ_1 posouvá registr instrukcí ke konečné mikroinstrukci LDA, a to je 00001001; ta dodává operační kód další instrukce v programové paměti do registru instrukcí.

Všechny kroky, které jsou nezbytné k vykonání instrukce LDA, se zdají být na první pohled poněkud složité. Projdete-li si však zběžně znovu předchozí odstavce, uvidíte, že instrukce LDA, stejně jako všechny instrukce PIP-2, je pouhou sbírkou velmi jednoduchých operací pospojovaných úhledně dohromady impulsy Φ_1 a Φ_2 z generátoru hodinových impulsů. Na obr. 31 jsou diagramy, které nám ukazují přesně, co se při čtení mikroprogramů děje.

Závěrem o řízení

Nyní, když jste se seznámili s tím, jak PIP-2 vyvolává, dekóduje a vykonává instrukci, snad teprve oceníte jeho vyspělou organizaci řízení. Řízení, jako takové, si můžete přirovnat k jednoduchému mikroprocesoru uvnitř PIP-2. Řídicí paměť ROM obsahuje program, registr instrukcí slouží jako čítač programů a dekodér mikroinstrukcí vykonává různé instrukce. Následující tabulka shrnuje mikroprogramy nezbytné k vykonání každé instrukce, která je v programovém instrukčním počítači PIP-2 použita.

Vedle mnemotechnických kódů (MK) a operačních kódů (OK) obsahuje tabulka úplnou pravdivostní tabulku řídicí paměti ROM a také ukazuje operace, které se uskutečňují pro každou mikroinstrukci.

Mikroprogramování pro PIP-2

Podívejme se na tabulku mikroprogramů. Všimněte si, jak často se vyskytují vyvolávací operace PC/I a RAM/R→IR/W. Odstraňte tyto mikroinstrukce z tabulky a zbude jen pět dodatečných mikroinstrukcí. Zřejmě existuje více možných mikroinstrukcí, než jenom těch sedm. Novou instrukci získáme vlastně tím, že určíme jeden zdroj a jedno nebo několik míst určení na adresovou a datovou sběrnici. Zde jsou některé z možností:

A/R→IR/W
A/R→PC/W
B/R→IR/W
B/R→PC/W
B/R→A/W
RAM/R→B/W
ADD/R→B/W
ADD/R→PC/W
ADD/R→IR/W

Samozřejmě, že toto jsou jenom některé z dodatečných mikroinstrukcí, které jsou možné. Všechny možnosti se nám otevřou, když bychom uvažovali o několika místech určení. Například RAM/R→A/W; B/W; PC/W, atd.

Jestliže se rozhodnete pro stavbu PIP-2, můžete nahradit mikroinstrukce svým vlastním souborem instrukcí, který si sami vytvoříte.

Předpokládáme-li např., že chcete nahradit instrukci LDA instrukcí LDB (naplnit registr B), všechno, co musíte udělat, je najít mikroprogram instrukce LDA v řídicí paměti ROM a přeprogramovat byte, který je v registru A (adresa 0011) tak, aby se místo registru A naplnil registr B. Původní byte je 00001010. Nový byte je 00001011. Zbývající byty zůstávají nezměněny. Operační kód pro instrukci LDA se tak stane operačním kódem pro instrukci LDB, protože jsme nezměnili lokalizaci mikroprogramu v řídicí paměti ROM.

Stejného postupu můžete využít k mikroprogramování dalších nových instrukcí pro PIP-2. Jen si zapamatujte tyto body:

1. Ujistěte se, že jste přidělili správný operační kód každé nové instrukci. Pamatujte si, že operační kód je dvojkové číslo, které

Programová paměť		Řídicí paměť ROM mikroprogramy				Činnost
MK	OK	adresa	OK	zdroj	určení	
NOP	1111	0000	01	000	000	PC/I
		0001	00	001	001	RAM/R→IR/W
LDA	0001	0010	01	000	000	PC/I
		0011	00	001	010	RAM/R→A/W
		0100	01	000	000	PC/I
		0101	00	001	001	RAM/R→IR/W
ADD	0101	0110	00	100	010	ADD/R→A/W
		0111	01	000	000	PC/I
		1000	00	001	001	RAM/R→IR/W
JMP	1000	1001	01	000	000	PC/I
		1010	00	001	100	RAM/R→PC/W
		1011	00	001	001	RAM/R IR/W
MOV	1011	1100	00	010	011	A/R→B/W
		1101	01	000	000	PC/I
		1110	00	001	001	RAM/R→IR/W
HLT	1110	1111	11	000	000	C/D

je o 0001 menší než první adresa mikroprogramu v řídicí paměti ROM.

2. Je-li nezbytné, zahrňte příslušné vyvolávací mikroinstrukce do každého nového mikroprogramu tak, aby příští instrukce v programové paměti byla vyhledána.

3. Ujistěte se, že mikroinstrukce na adrese 0001 v řídicí paměti ROM je vždy 00001001. Je to nezbytné proto, že tato mikroinstrukce hraje klíčovou roli ve vyvolání první instrukce z programové paměti během automatické začáteční sekvence PIP-2.

4. Plánujte dopředu! Nevylučujete nějakou existující instrukci, kterou byste mohli později potřebovat? Má řídicí paměť ROM dost místa pro nové instrukce? Máte dost programovacích zkratek, které můžete použít k vyplňování instrukcí nikoli v řídicí paměti ROM?

5. Dokumentujte si pozorně své úvahy i práci, abyste věděli, co jste udělali a jak jste při tom uvažovali.

Nenechte se těmito několika radami odradit od toho, abyste se pustili do mikroprogramování PIP-2! Některé možnosti jsou velmi zajímavé. Tak například instrukce, která naplňuje programový čítač obsahy registru A dovoluje, aby se program větvil na nějakou adresu, určenou výsledkem sčítání. Tento postup se nazývá nepřímé adresování. Dává mikroprocesoru schopnost větvit se (skákat) k jedné nebo několika možným adresám v jeho programové paměti, které závisí na výsledcích předchozí operace. Zde je jeden možný mikroprogram s nepřímým adresováním, o němž jsme právě mluvili:

mikroinstrukce

operace	zdroj	určení	
00	100	100	ADD/R → PC/W
00	001	001	RAM/R → IR/W

Protože druhá mikroinstrukce tohoto programu je stejná jako druhá mikroinstrukce NOP v původním souboru instrukcí pro PIP-2, můžeme ji velmi snadno nahradit za NOP. Právě jsme přeprogramovali první adresu v řídicí paměti ROM na 00100100 a přidělili operační kód od NOP k nové instrukci.

Pro jednoduchost lze přidělit nové instrukci mnemotechnický kód. Protože tato instrukce je nepřímý skok, je jednou z možností přidělit mnemotechnický kód JMI. Mohli byste však být přísnější, protože se vám může vyskytnout i další instrukce pro nepřímý skok a tuto instrukci bychom museli označit jako JMI. Protože výše uvedená instrukce je vlastně instrukce „Přeskoč nepřímo k adrese uvedené v registru A", byl by mnohem výhodnější mnemotechnický kód JIA.

Nyní, když víte, jak mikroprogramovat PIP-2, zkuste si sami přidat jednu nebo dvě instrukce. S trochou pozornosti se vám to určitě povede.

Závěr

Jestliže jste dočetli tento článek až sem, měli byste mít poměrně dobré znalosti o některých základních částech mikroprocesorů. K tomu, abyste se mohli pustit do stavby tohoto jednoduchého instrukčního počítače, musíte mít zkušenosti se zapojováním základních integrovaných obvodů (alespoň ve stavebnicích). Je nutno ještě říci, že skutečné mikroprocesory jsou mnohem složitější než PIP-2, popis činnosti PIP-2 vás však připravil k tomu, abyste se nyní mohli pustit do stavby podle následujícího návodu, nebo do stavby skutečných mikroprocesorů.

Literatura

Syrovátko, M.; Černoch, B.: Zapojení s integrovanými obvody. SNTL: Praha 1975.

Sobotka, Z.: Základy číslicové techniky, TESLA VÚST, ČVTS: Praha 1972.

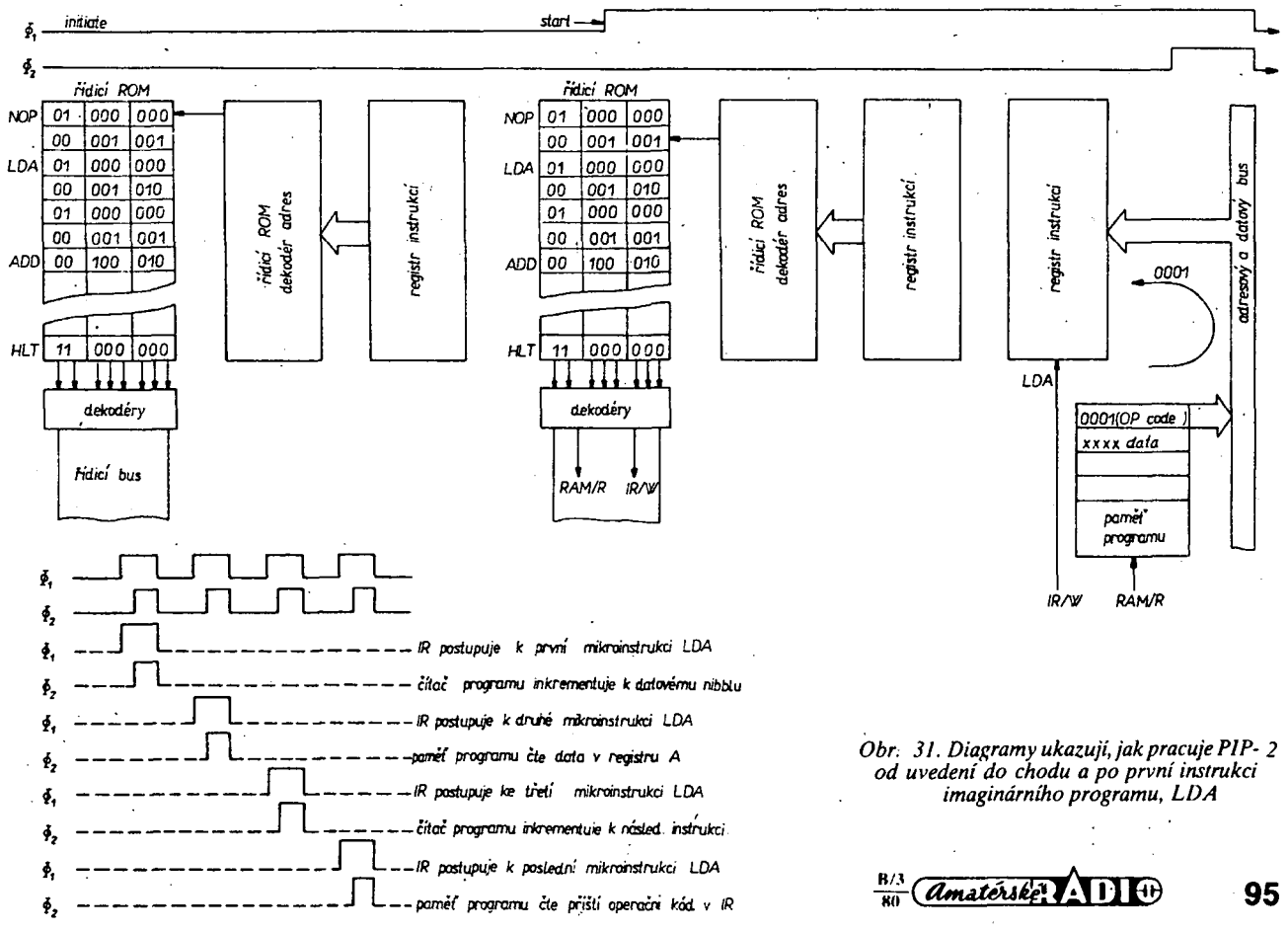
Mikrokurs o mikroprocesorech. Popular Electronics 1978.

Realizace počítače PIP-2

Jan Mercl

V následujícím textu je stručný popis řešení počítače PIP-2. Všechny obvody byly ověřeny při zhotovení jednoho kusu tohoto zařízení, které bylo realizováno v Městské stanici mladých techniků v Praze. Autorem elektrického obvodového řešení je Jan Mercl, člen kroužku kybernetiky, kterému bylo v době návrhu 16 let.

Autora technického řešení PIP-2 jsme požádali, aby se podílel na realizaci tohoto čísla AR. Charakteristická pro jeho přístup k věci byla první slova, která v této souvislosti pronesl: „Dobrá, avšak upozorňuji na to, že použitá zapojení nemusí být nejlepším možným řešením."



Obr. 31. Diagramy ukazují, jak pracuje PIP-2 od uvedení do chodu a po první instrukci imaginárního programu, LDA

Popis zapojení

Zapojení až do úrovně blokového schématu bylo vysvětleno v předchozím textu, v němž byly také definovány nejdůležitější signály a v podstatě vyřešeno zapojení řídicího bloku (ČB). V realizovaném zapojení (na obr. 32) jsou proti teoretickému návrhu různé změny. Bude na to dále upozorněno.

Řídicí blok (CB)

Řídicí blok je tvořen obvody instrukčního registru (IR), registru řídicího bloku (REG CB), řídicí pamětí ROM (CROM), dekodérem zdroje (SOURCE DECODER), dekodérem určení (DESTINATION DECODER) a přídavnými obvody pro generování zápisových a jiných impulsů v závislosti na hodinových impulsech.

Řídicí blok začíná registrem IR, který je tvořen integrovaným obvodem MH74193, do něhož se zapisuje informace ze sběrnice při IRW = H (H – vysoká úroveň, čili log. 1) a současně při Φ_2 = H. IRW. ORW . Φ_2 zabezpečuje H_9. Výstupní signál H_9 je aktivní v úrovni L (Low – nízká úroveň, log. 0). Každá náběžná hrana Φ_1 inkrementuje IR. Instrukční registr (IR) je nulován signálem INIT s úrovní H. Obsah IR se přepisuje do REG CB při Φ_1 = L. Obsah CB se stává adresou mikroinstrukce v CROM. Registr CB je tvořen integrovaným obvodem MH7475. Tento registr je potřebný pro stabilizaci adresy v době inkrementování IR a během zápisu do něj.

Řídicí paměť ROM je realizována integrovaným obvodem MH74188, tj. 32× 8 bitů a je využita z jedné poloviny. Druhá polovina zbývá uživateli k možné změně mikroprogramů, jak již bylo popsáno. Pokud je třeba adresovat druhou polovinu C ROM, odpojí se adresový vstup E od země (lze použít i přepínač) a připojí se na +5 V přes odpor 1 kΩ.

Dekodéry zdroje a určení. Mikroprogramová binární instrukce z řídicí paměti ROM je těmito dekodéry převedena na tvar ODKUD →KAM. Výstup příslušející konkrétnímu místu má úroveň L, jestliže se toto místo stává zdrojem nebo místem určení.

V CB je oproti předešlému výkladu změna v tom, že není použit dekodér operace. Pro tuto realizaci by byl zbytečný. Bit 7 určuje přímo inkrementování PC a bit 8 přímo nastavuje požadavek zastavení hodin úrovní H. Z tohoto důvodu je i změna v obsahu CROM na adrese F_{16}, v operační části mikroinstrukce je OP = 1 0 (tj. nastavený osmý bit = požadavek zastavení hodin) oproti OP = 1 1, které vyžaduje dekódování.

Za dekodéry jsou do linek zařazeny negátory N_5 až N_{11}, aby výběrová linka byla aktivní při H.

Hodiny a jejich ovládání. Do bloku generátoru hodinových impulsů patří oscilátor (integrovaný obvod 555, který je možno nahradit některým z osvědčených zapojení), dva klopné obvody D (MH7474), hradla H_1 až H_9 a negátory N_1, N_2 a N_{12}.

Integrovaný obvod 555 byl v této konstrukci použit z několika důvodů. Vyhovuje mu napájecí napětí +5 V, jeho výstup je slučitelný s TTL a změnou jednoho prvku v časovacím obvodu lze v širokých mezích měnit jeho kmitočet. Trimrem nastavíme kmitočet oscilátoru a tím i rychlost chodu celého počítače. Při pomalém chodu počítače můžeme demonstrovat jeho funkci a zároveň komentovat průběh vykonávání instrukcí.

Kmitočet oscilátoru je dělen dvěma v klopném obvodu D, zapojeném jako dělič. Z negovaného výstupu dostáváme přímo negovaný signál Φ_1. Vynásobením Φ_1 se signálem oscilátoru (s otočenou fází) získáme signál Φ_2. K_1 je stavový klopný obvod hodin. Je nastavován a nulován signály START HODIN L a STOP HODIN L. Výstup tohoto klopného obvodu povoluje nebo blokuje generování signálů Φ_1 a Φ_2. Signál START HODIN L je upravený stav tlačítka START. V klivové poloze tohoto tlačítka je jeden výstup H_1 uzemněn a tím je klopný obvod R-S (hradla H_1 a H_2) nastaven tak, že výstup hradla H_1 je ve stavu log. 1, tj. je povolen start počítače. Stiskneme-li tlačítko START, bude na vstupu hradla H_1 stav H a na vstupu N_1 bude stav L; na na výstupu N_1 bude úroveň H, která se „násobí" v hradle H_3 s povolením startu. Jestliže má být povolen start, musí být stisknuto tlačítko START, tj. výstup N_1 = H, pak výstup hradla H_3 bude ve stavu L, čili START HODIN L a klopný obvod K_1 se nastaví tak, že budou generovány impulsy Φ_1 a Φ_2. START HODIN (s úrovní L) však okamžitě nuluje klopný obvod R-S, tudíž na výstupu hradla H_1 bude úroveň L a start tedy není povolen. Start bude povolen až po vrácení tlačítka START do klidové polohy.

Signál STOP HODIN, který je aktivní při L, je signál zastavující generování hodinových impulsů Φ_1 a Φ_2. Smí se objevit jen za určitých podmínek. Pokud blok řízení (CB) vydá požadavek na zastavení hodin (H), pak při nejbližším hodinovém impulsu Φ_2 bude mít výstup hradla H_5 úroveň L a na výstupu hradla H_4 bude tedy bez ohledu na druhý vstup úroveň H. Negátor za hradlem H_4 upraví tento stav na úroveň L a hodiny se zastaví.

To byla jedna možnost. Nyní se podívejme na další případ, kdy je STOP vyžádáno uživatelem. Po stisknutí tlačítka STOP se nastaví klopný obvod R-S (hradla H_7 a H_8) tak, že na výstupu hradla H_8 bude úroveň H. Když pak bude mít úroveň H také Φ_2 a zároveň signál IRW, bude na výstupu hradla H_9 úroveň L a na výstupu N_2 bude úroveň H. Negovaný součin H_8 . N_2 bude L a na výstupu hradla H_4 bude úroveň H bez ohledu na stav druhého vstupu. Dále je to stejné. Ruční STOP se tedy provede až při nejbližším zápisu do instrukčního registru IR, tedy po natažení operačního kódu instrukce následující za instrukcí, během níž bylo stisknuto tlačítko STOP. Tato instrukce se neprovádí. V instrukčním registru je proto, aby po znovuodstartování mohl počítač pokračovat v přerušeném programu.

Nutno ještě podotknout, že hodiny se zastavují též při signálu INIT, který má úroveň L.

Programový čítač. Programový čítač je tvořen integrovaným obvodem MH74193. Zapisuje se do něj adresa další prováděné instrukce. Má záchytný registr pro stabilizaci dat ze sběrnice během zápisu do programového čítače. Důvod: při zápisu do programového čítače z paměti RAM se mění adresa a reakcí na to je i změna dat z paměti RAM, a tedy nedefinovaný zápis do programového čítače. Programový čítač se inkrementuje buď příkazem z řídicího bloku (CB), nebo ručně — stisknutím tlačítka SST (singl step).

RAM. Paměť RAM je reprezentována integrovaným obvodem MH7489. Údaje se do paměti RAM zapisují stisknutím tlačítka LOAD. Tato paměť RAM neguje informaci, což je odstraněno negujícím budičem sběrnice (4× MH7403).

Registr A a B. Oba registry jsou z integrovaných obvodů MH7475. Informace se zapisují z řídicího bloku při Φ_2 = H. Čtení je možné pouze z registru A, u něhož jsou negované výstupy spojeny přes negující čtveřici hradel obvodu MH7403 s datovou sběrnicí.

Sčítačka. Integrovaný obvod MH7483 neustále sečítá obsahy registrů A a B. Při čtení ze sčítačky do registru A musí být informace stabilní, aby se zpětně nezměnil součet a tím i obsah registru A. Registr sčítačky je blokován signálem AWL.

Ovládání a programování. Po zapnutí počítače je mnoho obvodů v nedefinovaném stavu. Abychom s ním mohli spolupracovat, musíme ho nejdříve uvést do žádaného stavu. Stiskneme tlačítko INIT, tím se nuluje programový čítač (PC), instrukční registr (IR) a zastavují se hodiny. Rozsvítí se LED WAIT (čekám). Nyní můžeme vkládat nebo opravovat program, popřípadě definovat z panelu některá důležitá paměťová místa nebo stavy čítačů. Chceme-li vkládat program, musíme stisknout tlačítko LOADING, tím se rozsvítí LED Loading a signalizuje, že jsme se napojili na datovou sběrnici a můžeme ukládat nová datová slova a instrukce. Tyto informace se vkládají čtyřmi tlačítky, označenými INPUT (vstup). Stisknutím tlačítka nastavíme log. 1, je-li tlačítko ve výchozí poloze, nastavujeme stav log. 0. Pro kontrolu je vstupní slovo zobrazeno na panelu čtyřmi diodami LED v bloku INPUT. Chceme-li datové slovo uložit jako instrukci programu na adresu, která je momentálně v programovém čítači, stiskneme tlačítko LOAD a programové slovo se zapíše do paměti RAM a zobrazí na dalších čtyřech diodách LED, umístěných na panelu v bloku RAM. Potom inkrementujeme programový čítač stisknutím tlačítka SST. Takto můžeme vložit a prohlížet celý program.

Datové slovo můžeme také vložit přímo do registru A nebo do programového čítače stisknutím tlačítka AW nebo PCW. Vložené datové slovo se zobrazí v bloku registru A, nebo v bloku PC.

Po vložení programu vypneme tlačítko LOADING. Chceme-li nyní nechat počítač pracovat podle vloženého programu, musíme nejdříve splnit podmínky: v instrukčním registru musí být před spuštěním programu od adresy A stav log. 0. Je-li programový čítač nastaven na A, nebo je-li operační kód instrukce na adrese A, v případě že je programový čítač na adrese A – 1 a program má začít na adrese A, v programovém čítači musí být 1111_2 (což je F_{16} v šestnáctkové soustavě).

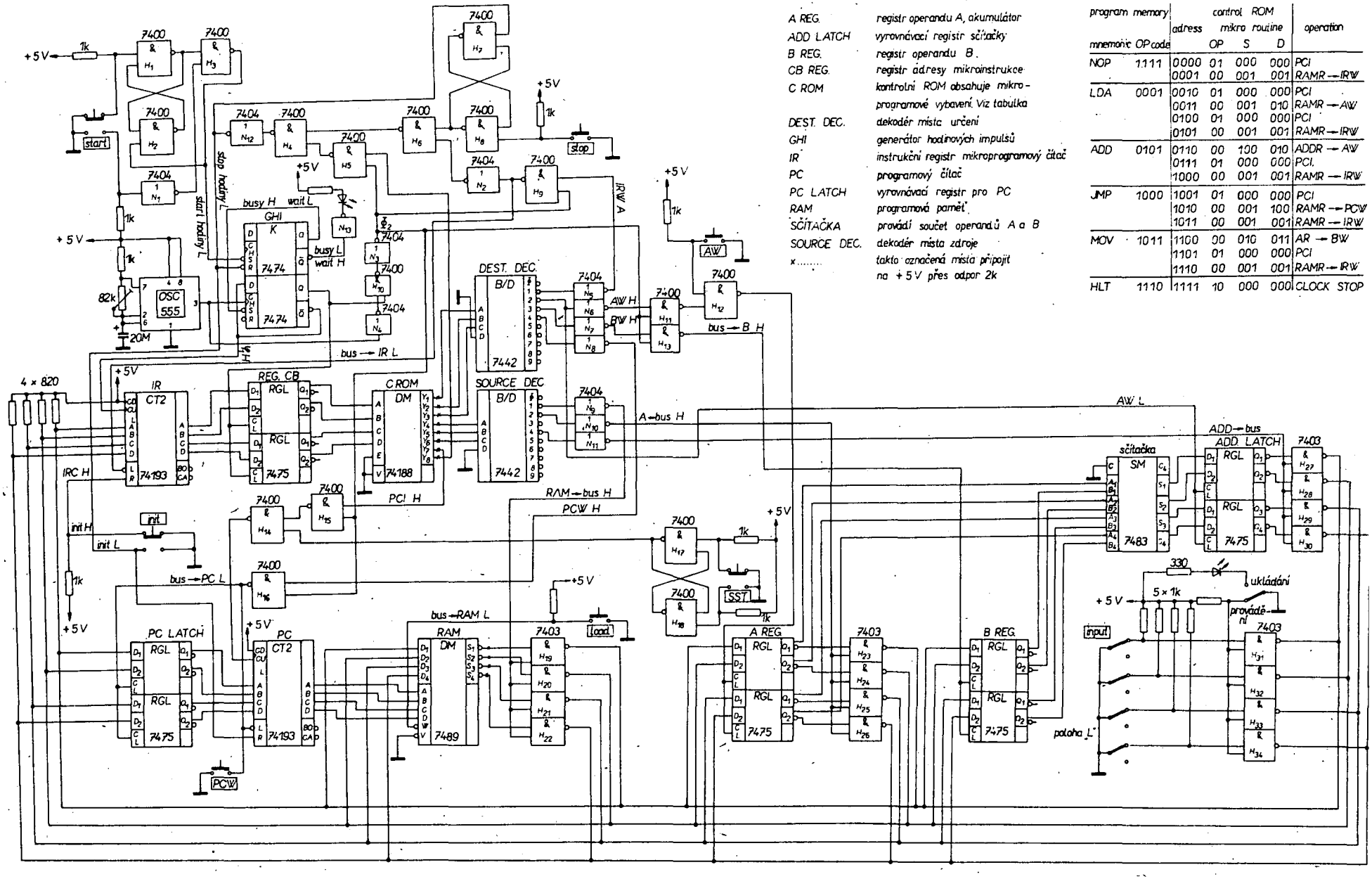
Instrukční registr vynulujeme stisknutím tlačítka INIT. Nemá-li program začínat od adresy 0_{16}, definujeme stav programového čísla vstupními tlačítky a tlačítkem PCW nebo tlačítkem SST. Operační kód následující instrukce se ocitá v instrukčním registru automaticky po ručním zastavení během programu.

F_{16} se v instrukčním registru ocitá po programovém zastavení instrukcí HLT. Protože v programovém čítači je na adrese A – 1, je F_{16}. Ovšem F_{16} je operační kód instrukce NOP a ta způsobí provádění programu až do adresy A.

Spouštění. Po definování všech míst a registrů v instrukčním registru a programovém čítači spustíme program stisknutím tlačítka START. PIP-2 má tak zvanou ochranu startu, to znamená, že nelze spustit program, jestliže již došel během stisku tlačítka START k instrukci HLT. Strojové HLT má větší prioritu. To je zabezpečeno tím, že po stisknutí tlačítka START se nastaví požadavek startu a tam se ruší ihned po jeho provedení; další požadavek je možný teprve po vrácení tlačítka START do klidové polohy. Pro provádění programu je nutné, aby v okamžiku spuštění nebylo stisknuto tlačítko LOADING (vkládání).

Zastavení. Počítač PIP-2 lze kdykoli požádat o zastavení běžícího programu:

a) stisknutím tlačítka INIT se plnění programu okamžitě zastaví,
b) stisknutím tlačítka STOP se zastaví, jakmile dokončí právě prováděnou instrukci. Při nejpomalejším chodu to může být i několik sekund, ale tlačítko STOP není



Obr. 32. Skutečné zapojení mikropočítače PIP-2

nutné držet po celou dobu. Stačí krátký stisk, který nastaví požadavek zastavení. Tento požadavek si klopný obvod „pamatuje" a provede ho, až jsou splněny výše uvedené podmínky.

Po zastavení programu, ať už ručním a nebo programovém, je možno spustit programové instrukce běžným způsobem, bez jekékoli definice čehokoli. K dalšímu pokračování v programu stačí stisknout tlačítko START.

Programátor ústředního topení

Ing. Eduard Smutný

Když jsme začali připravovat toto číslo RK, dohodli jsme se na klasickém uspořádání jednotlivých částí, tak abychom postupně vysvětlili, jak pracovat s integrovanými obvody, jak navrhovat kombinační a sekvenční logické obvody atd. a na závěr jsme chtěli dát konkrétní aplikaci a stavební návod na programátor ústředního topení. Když jsem však začal psát, došel jsem k názoru, že je třeba původní uspořádání poněkud změnit, abych neopakoval to, co již bylo publikováno jindy, popř. jinde.

Mnohdy je zapotřebí hodně hledat a listovat, než člověk najde to, co potřebuje a většinou navíc obvod nebo zapojení je třeba pro dané použití upravit. Já se zabývám navrhováním logických obvodů pro minipočítače a jejich přídavná zařízení již více než 10 let a ze svých zkušeností mohu říci, že se při návrhu opakuje několik standardních zapojení (derivační obvody, tvarovače apod.), která se mění pouze časem (např. monostabilní obvod z hradel se přestane používat, jsou-li k dispozici obvody 74121 nebo 123). Je proto třeba vypracovat si svůj vlastní postup návrhu obvodů a časem ho zdokonalit tak, aby byl rychlý a spolehlivý. Řešíme-li nějaký problém z číslicové techniky, zjistíme, že vlastní logický návrh není ani nejtěžší, ani nejpodstatnější částí řešení problému. Uvážíme-li, že použité obvody jsou dány tím, co je k dispozici v ČSSR, pak vlastní logický návrh od okamžiku, kdy víme přesně co chceme, může dělat nezkušený i zkušený návrhář – výsledek se nebude lišit co do počtu obvodů o více než 20 až 30 %. Větší rozdíl v návrhu bude patrný až tehdy, podíváme-li se na obě řešení z hlediska spolehlivosti celého zařízení, odolnosti proti rušení a ověříme-li si spolehlivost návrhu provozem zařízení a vliv tolerancí parametrů součástek zhotovením několika kusů. Pak se teprve objeví drobnosti, na které jsme zapomněli, nebo které se nám zdály nepodstatné. Proto je nutné při návrhu znát základní pravidla spolehlivého návrhu zařízení číslicové techniky a dodržovat je. Nejlepším zdrojem poučení pro návrháře je dokumentace, ať už našich nebo zahraničních zařízení, počítačů, přídavných zařízení atd. Dostane-li se vám taková dokumentace do rukou, pak je dobré ji podrobně prostudovat a všimnout si hlavně i zdánlivých maličkostí, týkajících se třeba blokovacích kondenzátorů na linkách napájecího napětí, ošetření vstupů a výstupů z logiky, způsobu vynulování klopných obvodů po zapnutí atd. Stejným způsobem budete pak v budoucnu postupovat při studování programů pro mikroprocesory, kde je třeba sledovat hardware i software najednou, aby bylo jasné, jak zařízení funguje a proč bylo děláno tak, a ne jinak.

Jak jsem již uvedl je třeba, aby si každý, kdo navrhuje číslicová zařízení, osvojil určitý způsob práce a ten postupně vylepšoval na základě nových poznatků a minulých chyb. Proto jsem se rozhodl napsat následující článek o realizaci programátoru ústředního topení tak, abych ukázal postup celé práce při návrhu, konstrukci, zhotovení a oživení číslicových zařízení. Programátor samozřejmě neobsahuje všechny obvody a problémy, které se vyskytují v číslicové technice, ale mohu z vlastní zkušenosti říci, že stejným postupem se navrhují počítače, řadiče flopydisků i mikroprocesorová zařízení.

Zadání úkolu

Zadání úkolu je první fází řešení každého zařízení. Ať už je zadavatelem podnik nebo kamarád, nebo si zadávám úkol sám pro sebe, je vždy důležité dohodnout se přesně na požadovaných funkčních vlastnostech zařízení, na podmínkách, v nichž bude pracovat, na instalaci, ceně nebo složitosti zařízení, způsobu napájení, obsluze atd. Mně byl úkol navrhnout programátor zadán kamarádem, který přišel s tím, že si postavil domek s ústředním topením na naftu a že má problém. Hořák topení je ovládán elektricky a je možno ho kdykoli zapnout nebo vypnout. První zimu zapínal hořák ručně s tím, že byl hořák ještě zapínán a vypínán termoregulátorem, nastaveným na požadovanou teplotu místnosti. Problém byl však v tom, že když chtěli odjet např. na hory, bylo nutno buď na tři dny topení vypnout a pak se v neděli večer vrátit do studeného domu, nebo po celé tři dny temperovat dům na přijatelnou teplotu. Stejný problém byl, jak topit, když jsou všichni v práci, nebo v noci. Určitým řešením by byly spínací hodiny, ale jejich programovatelnost je velmi omezená. Po ročním provozu byla spotřeba nafty shledána neúměrnou a tak přišel s tím, abych mu navrhl počítač, který by řídil topení celý týden podle kdykoli měnitelného programu. Dohodli jsme se po delší diskusi na základních parametrech zařízení:

a) základní programovatelnou jednotkou bude jedna hodina. To znamená, že během celé jedné hodiny bude hořák zapnut nebo vypnut, samozřejmě s tím, že regulační smyčka přes termostat zůstane v činnosti;

b) programovat bude možno vždy celý týden, to je 7 dní po 24 hodinách neboli 168 hodin. Příští týden může pak obvykle program zůstat stejný nebo bude možno naprogramovat nový týdenní program;

c) zařízení bude umístěno v obývacím pokoji, napájení bude však zajištěno z kotelny;

d) pro ovládání hořáku je třeba jeden spínací kontakt relé;

e) na zařízení bude možnost ovládat hořák z panelu, bez ohledu na program, aby nebylo nutno měnit program, když přijde návštěva, nebo když je v televizi náhodou dobrý pořad;

f) vypadne-li síť, nesmí se program porušit. Jako náhradní zdroj poslouží akumulátor 12 V, který bude současně sloužit jako zdroj pro měnič 12 V na 220 V pro napájení motorků topení a čerpadla;

g) na zařízení musí být nejméně dva indikační prvky. První bude indikovat, zda zařízení dává „dolů" informaci o zapnutí nebo vypnutí hořáku, a druhý bude indikovat, zda běží či neběží čerpadlo.

Potom jsem vzal sešit, na obálku jsem napsal HEATCOMP a na první stránku jsem napsal dohodnuté zadání úkolu. Tuto skutečnost uvádím tak podrobně proto, že pokládám za nezbytné psát vše, co se týká zařízení, do sešitu, a to od zadání až po oživení. Píše-li se na papír, stačí, aby se ztratil jeden list, a pak je třeba pracně objevovat již objevené. Sešity jsou pak v budoucnu neocenitelným pomocníkem, neboť určité problémy se neustále opakují. Do sešitu je vhodné psát všechno, co nás napadne, i když se nápad nezdá moc chytrý; na druhé straně na úpravě sešitu by nemělo záležet tolik, jako na technické přesnosti záznamů, ideální je ovšem jednota formy a obsahu.

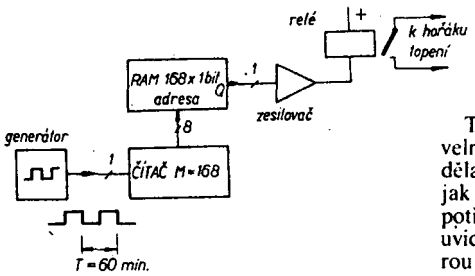
Úvahy o možném řešení

Mám-li zadání a sešit, mohu začít uvažovat, jak problém řešit. Obvykle v této fázi sháním literaturu, normy a dokumentaci obdobných zahraničních zařízení a přemýšlím. V tomto případě jsem udělal výjimku, neboť v literatuře jsem nic podobného neviděl; pokud jde o normy, pak na takové zařízení se vztahují pouze základní normy o bezpečnosti. O dokumentaci zahraničního zařízení jsem naštěstí nevěděl, neboť by mne zajisté roztrpčilo to, že by bylo na jediném čipu s odběrem několika mikroampér, zálohované baterií do hodinek a na schématu by byl jeden čtvereček a pár součástek. Technika, pracujícího v období mikroprocesorové revoluce, která ve světě nesporně probíhá, napadne ovšem řešení na bázi mikroprocesoru – před dvěma lety, kdy toto zařízení vznikalo, mne toto řešení napadlo též a musel jsem jej brzy opustit, protože potřebné obvody byly tehdy nedostupné. Nyní, kdy píši tento článek, je situace lepší a proto jsem přidal poslední kapitolu, v níž je ukázka přístupu k realizaci „Programátoru" na bázi mikroprocesoru INTEL 8080A. Čtenář má tudíž možnost porovnat oba způsoby řešení a případně si počkat, až budou tyto obvody dostupné na našem trhu s označením TESLA.

Vrátíme-li se zpět k úvahám o možném řešení programátoru z logických obvodů, pak jistě každého napadne, že základem bude generátor impulsů (krystalový oscilátor a děličky) s periodou $T = 60$ min.

Zde je třeba připomenout, že hodinové impulsy jsou v zařízeních s číslicovými obvody impulsy, které jsou přivedeny na hodinové vstupy čítačů a klopných obvodů, mluvíme-li o klopných obvodech, nebo také impulsy synchronizující chod části nebo celého zařízení, generované centrálním a stabilním multivibrátorem, nebo časovou základnou, mluvíme-li o zařízení jako celku. To, že v našem případě mají hodinové impulsy skutečně periodu jedné hodiny, je nutné brát jako náhodu a zároveň jako slovní hříčku.

Impulsy s hodinovou periodou budou čítány čítačem, který bude mít 168 stavů, 24 pro každý ze sedmi dnů v týdnu. Výstup čítače bude pak adresovat paměť RAM o kapacitě 168 slov. Délka slova je určena počtem informací které potřebujeme, v našem případě postačí 1 bit, neboť potřebná informace je pouze ZAPNUTO / VYPNUTO. Výstup paměti tedy udává přímo požadovaný stav kontaktů relé, které bude ovládat hořák topení. Blokové schéma řešení této části programátoru je na obr. 1. (Ve schématu je označeno přeškrtnutím a číslem, kolik vodičů spojuje jednotlivé bloky a šipkou směr toku informace. Tyto údaje jsou totiž velmi důležité pro pochopení činnosti a současně umožňují hned odhadnout, jak budou bloky složité; pomohou i při návrhu desky s plošnými spoji. Ten, kdo hned nepoznal, proč je mezi čítačem a pamětí RAM 8 vodičů, si musí spočítat, že binární čítač má 16 stavů, je-li 4bitový, 32 stavů, je-li 5bitový atd.)



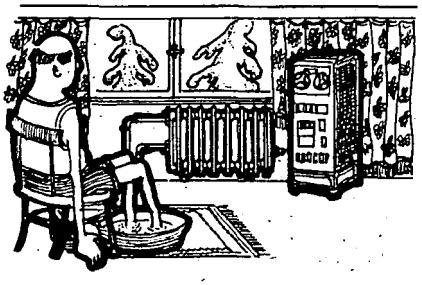
Obr. 1. Princip programátoru

Až potud je všechno jasné a můžeme začít uvažovat o tom, jak paměť naprogramovat, neboli jak do paměti RAM napsat, zda v příslušnou hodinu má být topení zapnuto (= log. 1) nebo vypnuto (= log. 0). Obr. 1 totiž odpovídá spíše použití paměti ROM než RAM, jak jistě některé čtenáře napadlo (a ještě přesněji paměti EPROM).

(Připomeňme si ještě pro úplnost, že uvedené uspořádání není jediné možné. Stejně dobře by bylo možno ukládat do paměti RAM 8bitová čísla, udávající hodinu, kdy se má topení přepnout z jednoho stavu do druhého, nebo dokonce pouze informace o tom, za jak dlouho se má topení přepnout, případně i s bitem, udávajícím stav v následujícím intervalu. Zde je možno najít obdobu v absolutním nebo inkrementálním programování polohy obráběcích strojů, nebo v tzv. modulaci delta. V prvním případě by stejná kapacita paměti umožnila pouze 3 až 4 sepnutí za den a v případě druhém by stačila délka slova 4 až 5 bitů, takže paměť by byla lépe využita, ale komplikovalo by se programování. Další úvahy již nechám čtenáři.)

V našem případě, kdy máme v každém slově informaci o zapnutí nebo vypnutí topení, je nutné naprogramovat paměť celou, bit po bitu a při změně programu pouze ty části, v nichž se bude program lišit. Při zápisu je tedy nutné projít všech 168 adres paměti a zapsat na tyto adresy 1 nebo 0. Pro adresování paměti můžeme použít stejný čítač, který adresuje paměť při čtení z paměti. Abychom zařízení nekomplikovali, budeme při programování čítat do čítače impulsy odvozené z tlačítka na panelu zařízení. Toto tlačítko si označíme +1 a jeho základní funkcí bude přičíst jedničku do čítače při každém stisku. Abychom mohli do paměti nejen zapisovat, ale i kontrolovat zapsaný program, zavedeme si ještě na panelu přepínač RUČ./AUT., který bude přepínat vstup čítače na impulsy z tlačítka + 1 nebo na impulsy o periodě 1 hodina. Dále si zavedeme přepínač ČT./ZÁP., který bude určovat, zda chceme do paměti zapisovat, nebo obsah paměti číst. Poslední informací, kterou při programování musíme zadat, je požadovaný obsah paměťové buňky – log. 1, má-li být topení v příslušné hodině zapnuto, nebo log. 0, má-li být vypnuto. Zapisovanou informaci volíme na panelu přepínačem I/O. Nyní, s vědomím, že jsme ještě mohli na něco zapomenout, se pokusíme znázornit graficky:

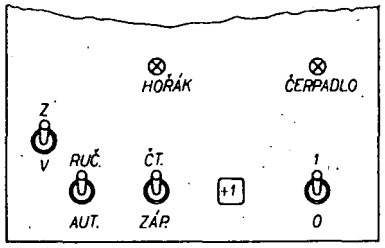
a) jak bude vypadat panel zařízení,
b) jaká bude funkce zařízení,
c) jaká bude obsluha zařízení,
d) blokové schéma zařízení.



Tyto body jsou pro další návrh zařízení velmi důležité. Ještě jsme totiž nezačali nic dělat, takže není co zkazit a můžeme si ověřit, jak jednoduchá bude obsluha zařízení, co je potřeba ještě přidat na ovládací panel. Jak uvidíme později, grafické znázornění je dobrou pomůckou pro logický návrh zařízení a u zařízení řízených mikroprocesorem nebo mikroprogramem je základem pro psaní programů.

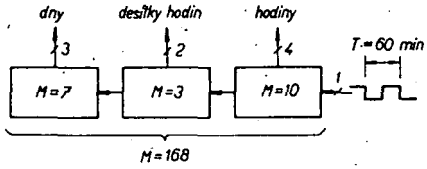
Ovládací panel

Na obr. 2 je předběžné rozložení ovládacích prvků na panelu programátoru. Bez ohledu na v budoucnu použité prvky, je třeba rozlišit tlačítka a přepínače. Panel si kreslíme proto, abychom při sestavování funkce a obsluhy zařízení mohli sledovat, jak se bude se zařízením pracovat a jak bude funkce záviset na jednotlivých ovládacích prvcích.



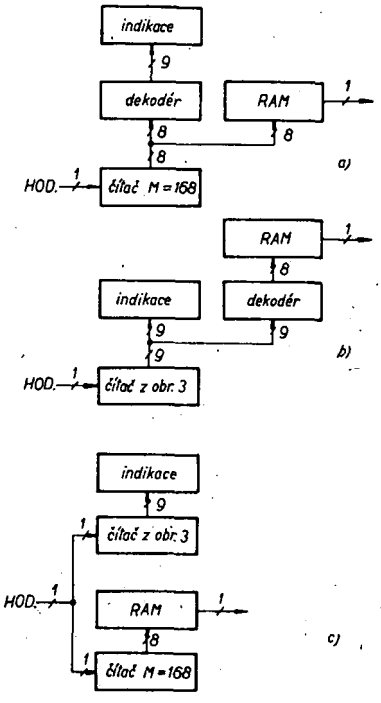
Obr. 2. Předběžné rozložení ovládacích prvků na panelu

Indikační prvky na panelu vyplývají ze zadání úkolu stejně tak, jako přepínač Z/V, který přímo ovládá relé hořáku. Začneme-li nyní přemýšlet o budoucí obsluze zařízení, brzy zjistíme, že na panelu chybí indikační nebo ovládací prvek, týkající se časové informace. Tak za prvé, chceme-li programovat nový program, bude vhodné, začneme-li od první hodiny prvního dne v týdnu. Po elektrické stránce to bude velmi jednoduché, protože stačí vynulovat čítač z obr. 1. Na panel proto přidáme tlačítko NUL. Nyní již by bylo možné programovat. Přepneme do polohy RUČ. a zvolíme ZÁPIS a tlačítkem NUL nastavíme čítač na nulu. Pak nastavíme přepínač I/O na požadovanou informaci, kterou chceme zapsat. Předpokládejme, že chceme, aby bylo topení vypnuto až do 4 hodin ráno, takže nastavíme 0. Pak stlačíme tlačítko +1. Stlačením tohoto tlačítka se zapíše 0 na adresu 00000000 paměti RAM a po zápisu se zvětší obsah čítače o jedničku. Při opětovném stlačení tlačítka +1 se zapíše 0 na adresu 00000001 a opět se inkrementuje čítač, takže stlačíme tlačítko ještě dvakrát a máme naprogramovány první čtyři hodiny. Pak chceme, aby topení bylo zapnuto až do 7 hodin ráno. Přepneme proto přepínač na 1 a stlačíme tlačítko +1 třikrát po sobě. Takto můžeme naprogramovat celých 168 hodin týdne. Po skončení programování je možno přepnout na ČTENÍ, tlačítkem NUL opět vynulovat čítač a program si zkontrolovat. A hned máme další problém, neboť nám chybí indikace výstupu paměti. Při režimu AUT poslouží k indikaci přímo žárovka HOŘÁK, ale při RUČ. je hořák ovládán přímo přepínačem Z/V a ne výstupem z paměti. Proto přidáme na panel ještě jednu žárovku a nazveme ji třeba PROG. – ta bude ukazovat naprogramovanou hodnotu pro danou hodinu. Takto řešené zařízení by funkčně vyhovělo, ale obsluha by si musela pamatovat, kterou hodinu právě programuje (nebo kterou čte) a při spuštění do automatického režimu by musela v neděli počkat do půlnoci a pak přepnout na AUT a stlačit tlačítko NUL, nebo by musela stlačit NUL a „odmačkat" při RUČNĚ a ČTENÍ počet hodin, které by uplynuly od první hodiny týdne a pak přepnout do AUT. Já jsem vyřešil tento problém tak, že indikuji časovou informaci na panelu přístroje. Čtenář může sám posoudit, co stojí takový přepych a případně si postavit programátor bez indikace času. Pro indikaci času jsem přidal na panel 7 žárovek pro dny v týdnu a dvoumístný číslicový displej pro 24 hodin jednoho dne. Současně s tím jsem ovšem musel začít hned přemýšlet, jakými obvody bude možné indikaci realizovat. Problém byl v tom, že vhodná paměť pro programátor měla kapacitu 256 slov × 1 bit (74S201), kdežto čítač, jehož stav by bylo možno indikovat na panelu, musí být uspořádán tak, jak je znázorněno na obr. 3.



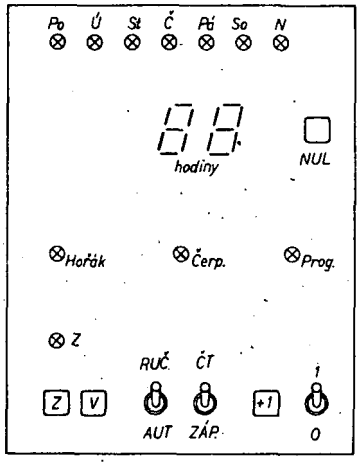
Obr. 3. Čítač pro indikaci hodin a dnů v týdnu

Z obrázku je důležité pouze to, že pro indikaci hodin potřebujeme 4 bity, pro desítky hodin 2 bity a pro dny 3 bity, to je celkem 9 bitů. Pro adresaci paměti však potřebujeme pouze 8 bitů. Možná řešení tohoto problému jsou na obr. 4. Řešení A vyžaduje dekodér osmibitového binárního čísla na 9 bitů, tento dekodér by byl velmi složitý a rozumně by šel realizovat pouze pamětí ROM o kapacitě 256 × 10 bit, u níž by bylo



Obr. 4. Řešení problému adresace a indikace

možno využít desátého bitu k nulování čítače tak, aby počítal pouze do 168 a ne do 256. Řešení B vyžaduje dekodér, který jednoznačně přiřadí každému ze 168 stavů čítače jednu adresu paměti RAM. Jelikož adres je 256, je zde určitá volnost v návrhu, takže dekodér by nemusel být složitý. Řešení C je jednoduché a někdy se objeví v literatuře nebo zahraničních dokumentacích pro převod kódu BIN a BCD při vstupu a výstupu z počítače (fy DEC USA). O konkrétním řešení však můžeme rozhodnout později, teď řešíme panel a je vidět, že indikace je schůdná. Při překontrolování postupu obsluhy zařízení podle obrázku panelu jsem přišel na to, že použít přepínač Z/V nebude nejvhodnější, neboť by se mohlo stát, že obsluha nechá tento přepínač omylem v poloze Z a topení bude trvale zapnuto a nebude řízeno programem. Proto jsem nahradil tuto funkci klopným obvodem Z, který je možno nastavit a vynulovat z panelu a navíc bude vždy o půlnoci nulován. Stav klopného obvodu pak bude na panelu indikován žárovkou Z, protože po nahrazení přepínače tlačítky již obsluha nepozná, zda si topení zapnula navíc proti programu. Po těchto úvahách je možno nakreslit doplněný panel (obr. 5).



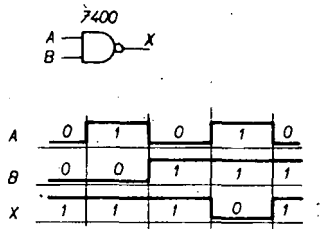
Obr. 5. Doplněný návrh panelu

Funkce zařízení

Funkci zařízení si můžeme buď popsat nebo znázornit graficky. Funkci graficky znázorňuji formou vývojových diagramů. Vývojové diagramy jsou přehledné a lze z nich vycházet ať navrhujete zařízení z běžných logických obvodů, nebo řízené mikroprogramem v paměti ROM, nebo řízené minipočítačem či mikroprocesorem. Velkou výhodou je, že se dají dělat na různých úrovních, obdobně jako blokové schémata, od názorných, zobrazující základní funkce, až do detailních, znázorňujících každou změnu stavu logických obvodů nebo případně každou instrukci mikroprocesoru. V této fázi je nejvhodnější zvolit nějakou střední úroveň, která nebude ještě závislá na konkrétním logickém řešení. Funkci zařízení nekreslím obvykle do jednoho vývojového diagramu, protože některé funkce začínají v zařízení asynchronně vzhledem k ostatním funkcím, takže by byly problémy časovacího rázu, tzn. že vývojový diagram by nepopisoval správně časový sled operací. Jak uvidíme dále, můžeme se dostat při sledování funkce zařízení do čekací smyčky, kdy čekáme třeba na puštění tlačítka +1 a současně je možno stlačit kdykoli tlačítko NUL a vynulovat čítač. Při programování mikroprocesorových zařízení je tento problém ekvivalentní zakreslení žádosti a obsluhy přerušení programu do vývojového diagramu probíhajícího programu. I zde je pak lepší rozdělit vývojové diagramy na jednotlivé asynchronní podiagramy.

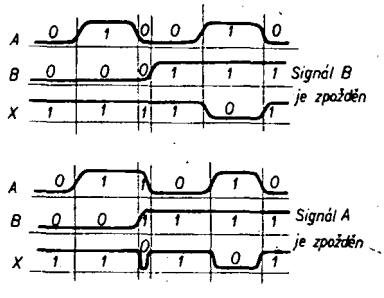
Zde je nutné mít již rozmyšleno, jak budou některé obvody realizovány. Tak například nulování čítače tlačítkem NUL nebo inkrementaci čítače tlačítkem +1 je možno řešit dvěma způsoby. Při asynchronním řešení je signál z tlačítka veden přímo na nulovací vstup čítače, takže stiskneme-li tlačítko, čítač se ihned vynuluje. Při synchronním řešení zařízení vyrábí časová základna společně s dalšími obvody zařízení hodinové impulsy a určitá činnost, třeba nulování čítače, pak může nastat pouze tehdy, je-li stisknuto tlačítko a generuje-li současně časová základna impuls T_n, který byl při návrhu zařízení časově umístěn tak, aby nulování čítače mohlo proběhnout v okamžiku, kdy to ničemu nevadí, případně kdy to přesně potřebujeme. Rozhodnout, zda zařízení nebo jeho části řešit synchronně nebo asynchronně vyžaduje určité zkušenosti – můžete vycházet z následujících úvah.

První problém, který lze odstranit synchronním řešením obvodů, jsou hazardní impulsy. O hazardních impulsech toho bylo napsáno mnoho, ví se o nich, a přesto se prakticky vždy v logických zařízeních objeví, alespoň v těch, které dělám já. Psávalo se, že hazardní impuls se může objevit na výstupu hradla nebo kombinačního logického obvodu, mění-li se logická úroveň na více než jednom vstupu tohoto obvodu – to už dnes neplatí. Paměť ROM, třeba MH74188, dává hazardní impulsy, i když se mění v dané chvíli pouze jeden adresový vstup. Není to ovšem její chyba, v příručkách firem je na to upozorněno a když to přehlédnete, pak hledáte hezky dlouho, než na hazardní impulsy sondou přijdete, osciloskop je na ně obvykle krátký. Hazardní impuls vznikne třeba na výstupu hradla NAND, mění-li se jeden jeho vstup z 0 do 1 a druhý z 1 do 0. Obvykle si člověk nakreslí časový diagram tak, jako na obr. 6 a všechno je v pořádku. Uvažujeme-li, že



Obr. 6. Časový diagram na obvodu 7400

signály A a B se nemusí měnit přesně ve stejný okamžik, pak si můžeme nakreslit časový diagram s přehnaným zpožděním jednoho nebo druhého signálu tak, jako na obr. 7 a vidíme, že v prvním

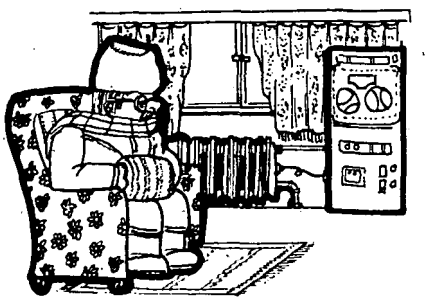


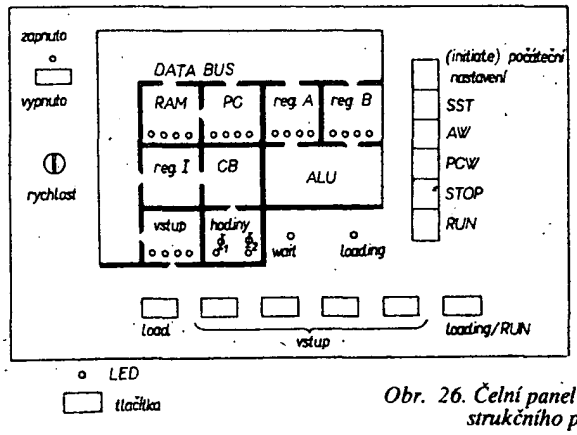
Obr. 7. Časové diagramy s uvažováním zpoždění

případě časový diagram souhlasí a v druhém případě je tam impuls navíc, obvykle dlouhý několik desítek nanosekund a to je ten hazardní šotek číslicových zařízení. Pozor na něj! Vzniká na výstupech obvodů 7442, dekódujeme-li jimi čítač, na výstupech multiplexů, na společných sběrnicích mikropočítačů atd. Bojujeme proti němu buď kondenzátorem (max. kapacita 1 nF) mezi výstupem a zemí obvodu (nestyďte se za to, dělá to i firma FACIT), nebo se používá třetí vstup, kterým hradlujeme neboli vzorkujeme výstup hradla v okamžiku, kdy už se zaručeně žádný vstup nemění, nebo třeba přepisem výstupu hradla do klopného obvodu D, opět ve správném okamžiku. Poslední dva způsoby jsou již vlastně v možnostech synchronních obvodů, kde třeba inkrementujeme čítač impulsem T_1, a po čase vyvzorkujeme dekodér čítače, třeba obvod 7442, časem T_2 a je po hazardech. Hazardním impulsům ovšem vděčíme za elegantní řešení derivačních obvodů, kde si zpoždění vytvoříme uměle, o čemž se čtenář může přesvědčit v Kuchařce.

Další výhody synchronního řešení vyplývají z toho, že časové impulsy jsou odvozeny z jednoho oscilátoru, ten může být i krystalový a různé časy potřebné pro práci zařízení jsou teplotně stabilnější a dají se měnit všechny najednou (třeba takt počítače). Někde je třeba právě proto obě řešení kombinovat, chci-li měnit jeden čas a chci-li, aby ostatní pracovaly synchronně a stabilně. Příkladem je třeba časová základna feritové paměti, řešená obvykle synchronně a v ní vzorkovací impuls okamžiku čtení, generovaný asynchronně, aby jej bylo možno nastavit. Někde se zase synchronní řešení nabízí samo, protože zařízení stejně potřebuje pro nějaký účel generátor impulsů a bylo by škoda ho nevyužít pro ostatní časování. Asynchronní řešení, které nemá pevné časy, má zase výhodu v tom, že se nemění stav několika obvodů v jeden časový okamžik a celkové rušení vznikající změnou odběru proudu obvodů při změně stavu je menší jak na „zemích", tak na napájecích vedeních. Asynchronně je nutné řešit obvody také tam, kde jsou požadované časy v širokých mezích (např. zpoždění 200 ms je lepší řešit monostabilním obvodem než čítáním kmitočtu třeba 1 MHz). Obvykle se v zařízení oba způsoby prolínají. Budete-li však navrhovat nějaké zařízení, řízené mikroprogramem, uloženým v paměti ROM, pak uvažte, že synchronní řešení má tu přednost, že se dají „hodiny" pustit pomalu nebo po krocích, usnadníte si tím oživování zařízení.

Při kreslení vývojových diagramů používám několik grafických symbolů. Na začátku diagramu ovál, do něhož vpisuji buď START nebo název funkce, kterou diagram popisuje. Kolečko v případě, že se diagram musí nakreslit na několik stránek. Do kolečka napíši číslo a na další stránce začíná pokračování diagramu opět kolečkem se stejným číslem. Obdélník, říkám mu výkonný, do kterého vpisuji, co se má v daném okamžiku vykonat. Protože je pole obdélníku malé, používám různé zkratky, grafické nebo písmenové, a je-li třeba, vysvětlivky píši vedle diagramu. Obdélník s tlustou čarou vlevo označuje výkonný obdélník, rozepsaný dále do samotného vývojového diagramu a označuje podfunkce nebo podprogramy – rutiny, které by zhoršily přehlednost hlavního vývojového diagramu. Kosočtverec pak označuje rozhodovací člen a má obvykle jeden vstup a dva, někdy i více výstupů. Do kosočtverce se píše podmínka a výstupy se označí podle jejího splnění nebo nesplnění. Jednotlivé grafické symboly se propojují čarami, které doplňuji šipkami určujícími směr „toků" nebo vývoje diagramu, obdobně jako u blokového diagramu. Při sestavování nebo sledování našeho vývojového diagramu je dobré mít před sebou obrázek ovládacího panelu. Tok vývojového diagramu našeho zařízení je značně závislý na stavu ovládacích prvků panelu a současně si při sestavování znovu ověříme budoucí obsluhu zařízení. Vývojový diagram funkce programátoru ústředního topení je na obr. 8. Vývojové diagramy rutin





Obr. 26. Čelní panel programovatelného instrukčního počítače PIP-2

Programová paměť

Jedná se o 64bitovou paměť RAM, která je organizována jako šestnáct čtyřbitových slov. Tato paměť RAM má tříhodnotový výstup, který slouží k tomu, aby její instrukce a data byly izolovány od sběrnice adresových dat, dokud jich není třeba.

Programová paměť má jednoduchý řídicí vstup, RAM/R (R = čtení). Je-li na řídicím vstupu RAM/R malá logická úroveň (log. 0), snímá paměť RAM slovo, které je adresováno programovým čítačem na sběrnici adresových dat. Je-li signál na RAM/R úrovně log. 1, mohou být vložena do paměti RAM instrukce a data.

Programový čítač

Je to čtyřbitový binární čítač. Mnohé skutečné mikroprocesory mají zvláštní paměťový adresový registr, který zabezpečuje obsahy programových čítačů, dokud není čas přistoupit k další adrese. Tyto adresové registry obsahují adresu té instrukce, která je právě realizována. Programový čítač však obsahuje adresu příští instrukce, která bude zpracovávána v následujícím cyklu. PIP-2 plní ve svém programovém čítači zároveň funkci adresového registru.

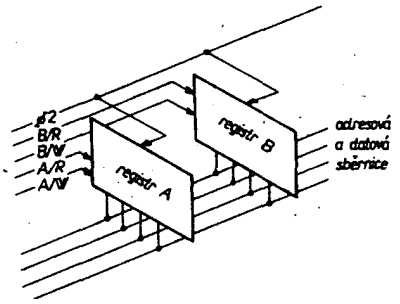
Programový čítač má tři řídicí vstupy. Vstup PC/C je aktivován spínačem INITIATE, úroveň log. 0 způsobí vynulování čítače na 0000. Vzestupná hrana impulsu přivedená na vstup PC/I způsobí inkrementaci (přírůstek) obsahu čítače. Obsah čítače se zvětší o jednu. Přivedeme-li na vstup PC/W sestupnou hranu impulsu, zaznamenají se do čítače jakákoli data, která jsou v tomto časovém okamžiku na sběrnici adresových dat – to je velmi cenné zařízení, neboť umožňuje větvit program, popř. vložit přes sběrnici jakoukoli adresu z programové paměti do programového čítače.

Registry A a B

Jsou to standardní čtyřbitové registry s tříhodnotovými výstupy. Každý z obou registrů má dva řídicí vstupy a hodinový vstup (Φ_2). Je-li na řídicím vstupu A/R (nebo B/R) úroveň log 0, jsou data snímána (read = číst) z vybraného registru A (nebo B) na adresovou a datovou sběrnici. Je-li na řídicích vstupech A/W a B/W úroveň log. 0, jsou jakákoli data z adresové a datové sběrnice zaznamenána na nějaký z vybraných registrů a to v tom okamžiku, kdy přijde další hodinový impuls (Φ_2) (obr. 27).

Sčítačka

Je to čtyřbitový kombinačně logický obvod, který kontinuálně sčítá obsahy registrů A a B. Součet je izolován od adresové



Obr. 27. Připojení registrů A a B k adresové a datové sběrnici a připojení řídicích vstupů pro čtení, zápis a hodinového vstupu Φ_2

a datové sběrnice tříhodnotovým vyrovnávacím zařízením. Je-li na řídicím vstupu ADD/ /R úroveň log. 0, je umožněno vyrovnání a součet je vložen na sběrnici.

Výstup

PIP-2 má výstup, který se skládá ze čtyř svítivých diod (LED), které trvale ukazují obsah registru B. Místo diod LED lze však k výstupu připojit vhodné vnější zařízení – tak například po připojení až šestnáctiřádkového dekodéru by PIP-2 mohl řídit až šestnáct vnějších zařízení.

Řízení

Řídicí sekce je elektronické nervové centrum našeho programového instrukčního počítače. Řízení dopravuje instrukce z programové paměti a vykonává jednu instrukci po druhé, podle přesně synchronizovaných řídicích signálů (Φ_1 a Φ_2), které vznikají v hodinovém (taktovacím) obvodu (clock).

Řídicí sekce se skládá ze 128bitové paměti ROM, organizované jako šestnáct osmibitových bytů, z dekodéru adres a dvoufázových „hodin". Jak jsme již uvedli, čítač programů v PIP-2 v sobě sdružuje funkci čítače mikroprogramů, a je tak těsně spjat s řízením, že může být považován za jeho součást.

Později se podíváme na blokový diagram řízení a podrobně prostudujeme jeho činnost. Zatím nám stačí, když si řekneme, že paměť ROM pro řízení obsahuje řadu 1 až 5 mikroinstrukcí pro každý z různých mikroprogramů, nezbytných k vykonávání šesti instrukcí PIP-2. Jak se jistě pamatujete, vykonávají mikroinstrukce jednoduché operace, jako například přenos dat z jednoho registru do druhého apod.

Soubor instrukcí pro PIP-2

PIP-2 může zpracovávat šest různých instrukcí. Každá instrukce je pro lepší zapamatování označována určitým druhem „těsnopisu", který nazýváme mnemotechnickým kódem – pro PIP-2 je to čtyřbitový nibble, označovaný jako operační kód.

Některé instrukce vyžadují pouze jednu adresu programové paměti, zatímco jiné jsou následovány datovým slovem. Tyto druhé instrukce vyžadují dvě adresy programové paměti a nazývají se paměťové referenční instrukce. Například

0001 (LDA)
1111 (data)

je formát pro paměťovou referenční instrukci, která vkládá do registru A (LDA) datové slovo 1111.

Jak je ukázáno v následující tabulce, nevyžaduje soubor instrukcí pro PIP-2 mnoho dalšího vysvětlování:

Soubor instrukcí pro PIP-2

mnemotechnicky	operační kód	nibbly	činnost
NOP	1111	1	žádná činnost
LDA (nibble)	0001 (xxxx)	2	vlož do registru A následující nibble
ADD	0101	1	sečti registry A + B; výsledek uchovej v A
JMP (adresa)	1000	2	přeskoč k adrese
MOV	1011	1	přemísti obsah
HLT	1110	1	zastav mikroprocesor

Používat tyto instrukce ve skutečných programech je snadné, pokud ovšem víme, jak a kdy je používat. Proto si dále výše uvedené instrukce probereme podrobněji jednu po druhé.

NOP (no operation). Je to instrukce pro „nicnedělání" s několika cennými aplikacemi. Jednoho nebo dvou NOP můžete používat k rezervování místa v programu pro jednu nebo dvě instrukce, které byste si mohli přát přidat k programu později (během ladění programu). Stejně tak je možno použít instrukce NOP, chceme-li odstranit některou z instrukcí, aniž bychom přepisovali program. A konečně můžeme NOP použít k přidání předpokládaného časového zpoždění k programu, což je výhodné při cejchování nějakého programu, který „kličkuje" nějakým cyklem instrukcí znovu a znovu a působí tak jako omezovač (spínací hodiny).

LDA (load accumulator). Tato paměťová referenční instrukce „zásobuje" registr A nibbly dat v příští adrese programové paměti. Používá se k dočasnému uskladnění nějakého nibblu pro dodatečný nebo pozdější přenos k výstupu nebo k čítači instrukcí.

ADD (addition). Tato instrukce pro jeden krok programu spouští řetěz pěti mikroinstrukcí, které přidávají obsahy registrů A a B a výsledek vkládají do registru A. Používá se pro obyčejné sčítání a přirůstání nibblu v registru A jako střadače (který najdeme také ve skutečném mikroprocesoru).

JMP (jump). Je to velmi významná instrukce, která nařizuje programovému čítači, aby větvil program, popř. aby „skočil" k adrese v programové paměti, která je určena v následujícím nibblu. JMP se používá k nastave-

ní skoku, programu nebo části programu, který pokračuje, a „skáče" se potud, dokud není PIP-2 zastaven tlačítkem STOP.

MOV (move). Tato instrukce pro přenos registru má několik možností použití. Jako výstupní instrukce dovoluje operátoru PIP-2, aby viděl obsah registru A na displeji – diodách LED. Dále dovoluje provádět ekvivalent instrukce LDB (pro plnění registru B) tím, že před ní předřadíme instrukci LDA. A konečně umožňuje zdvojnásobit nějaké číslo tím, že ho následuje instrukce ADD.

HLT (halt). Tato instrukce se dává na konec programů PIP-2. Vyřazuje „hodiny" v řídicí sekci z činnosti a tak zamezuje realizovat jakékoli dodatečné instrukce.

Později si povíme o mikroprogramech pro jednotlivé instrukce a také se naučíme, jak přidávat nové instrukce tím, že změníme mikroinstrukce v řízení paměti ROM.

Jak programovat PIP-2

Příklad: Napište jednoduchý program pro PIP-2, při němž se budou postupně zvětšovat čísla v registru A o jednu a nechte součet znázornit na výstupu – diodách LED. Zde je zmíněný program:

adresa programové paměti	mnemotechnika a data
0000	LDA
0001	0001
0010	ADD
0011	MOV
0100	JMP
0101	0000
0110	HLT

Jasně vidíme, jak PIP-2 zpracovává tento program. Je-li PIP-2 spuštěn, jsou jak registr A, tak registr B vynulovány na 0000. To znamená, že první tři instrukce vkládají 0001 do registru A, přičítají obsah registru A k obsahu registru B a výsledek (0001) uchovávají jak v registru A, tak v registru B. JMP znamená, že program „skáče" zpátky na řádku 0000 pro další cyklus. Do registru A bude znovu uloženo 0001, registr B obsahuje 0001 z minulého cyklu, takže pro splnění instrukce D je obsah registru A 0010. Výsledek, 0010, je instrukcí MOV „přestěhován" do registru B a znázorněn na displeji.

A znovu, JMP znamená, že program „skáče" zpátky na řádku 0000 a výpočetní pochod pokračuje. Výsledkem je, že se na displeji objeví binární výpočet v rozmezí 0000 až 1111 a opakuje se, dokud není PIP-2 zastaven.

Jak je vidět, není tento program ničím víc, než programovanou verzí (software) nějakého obyčejného čtyřbitového čítače. To však není na PIP-2 nic zvláštního, protože již ve své technické konstrukci (hardware) obsahuje dva takové čítače. Zvláštní však je, že tento jednoduchý program může být snadno změněn (modifikován), aby vykonával jakýkoli početní přírůstek od 0000 do 1111, a to jednoduše tím, že změníme datový nibble následující LDA! I když by toho mohlo být dosaženo nějakým relativně jednoduchým hardware, vykonává PIP-2 takový úkol pouze po několika sekundách modifikace programu. To ilustruje pozoruhodnou univerzálnost tohoto mikroprocesoru, který dovede napodobit mnoho hardwarových funkcí s pomocí software.

Chod programu

Jednoduchý program čítače, který diskutujeme, se nazývá výchozí program, protože je zapsán s použitím mnemotechnických symbolů různých instrukcí. Než může být vložen do programové paměti PIP-2, musí být přeměněn na cílový program. Cílový program se zapisuje dvojkovými čísly, kterým mikroprocesor „rozumí". Cílový program se někdy nazývá též program v jazyce stroje. Vše, co je potřeba k vytvoření cílového programu pro náš „software" mikroprogram, je nahradit příslušné mnemotechnické symboly operačním kódem podle tabulky, splňující soubor instrukcí pro PIP-2. Zde je výsledek:

adresa	výchozí program	cílový program
0000	LDA	0001
0001	0001	0001
0010	ADD	0101
0011	MOV	1011
0100	JMP	1000
0101	0000	0000
0110	HLT	1110

Když je cílový program sestaven, je už jednoduché vložit ho do programové paměti PIP-2. Nejprve zapneme napájení, tím se samočinně vynuluje programová paměť, registry a čítače. Potom spínačem na čelním panelu (spínač v dolní poloze odpovídá úrovni log. 0, v horní poloze úrovni log. 1) předvolíme první cílový kódový nibble v programu (0001) a stiskneme tlačítko LOAD. Tak se vloží nibble 0001 do adresy 0000 programové paměti a automaticky posouvá čítač instrukcí k další adrese.

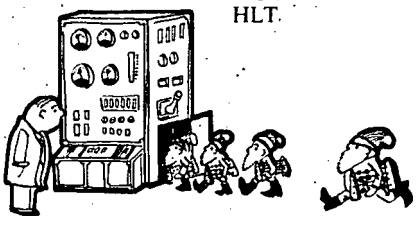
Zbývající nibbly jsou ukládány jeden po druhém do té doby, pokud nejsou všechny sekvenčně uloženy v programové paměti. Potom stiskneme tlačítko INITIATE, aby se čítač instrukcí vrátil na adresu 0000 v programové paměti. Nyní zbývá pouze stisknout tlačítko START. Řídicí sekce si připraví první instrukci z programové paměti, vloží ji do registru instrukcí, dekóduje ji a realizuje. Program je takto zpracováván krok po kroku tak, jak jsou znázorňovány obsahy registru B na displeji (diodách LED).

Je-li rychlost „hodin" (kmitočet hodinových impulsů) větší než asi 100 Hz, bude součet znázorněný na displeji s diodami LED rozmazaný do stále se opakujících 1111. Protože „hodiny" skutečných mikroprocesorů mají kmitočet řádu jednotek MHz i vyšší, musí být do jejich programů přidáno zpožďovací zařízení, aby znázorňovaná data mohla být operátorem přečtena.

Programy pro PIP-2

I když je soubor instrukcí pro PIP-2 velmi primitivní, lze s ním napsat určité množství různých programů. Tak např. je dále uveden výchozí program, který přidává dvě čísla a znázorňuje jejich součet:

```
LDA
(první číslo)
MOV
LDA
(druhé číslo)
ADD
MOV
HLT
```



A zde je výchozí program, který zdvojuje nějaké číslo:

```
LDA
(číslo)
MOV
ADD
HLT
```

A zde je program, který počítá po dvojkách:

```
LDA
0002
ADD
MOV
JMP
0000
HLT
```

Tím samozřejmě programové možnosti PIP-2 zdaleka nekončí.

Programování skutečného mikroprocesoru

Skutečný mikroprocesor má desítky instrukcí ve svých souborech instrukcí. Typický mikroprocesor (jako např. 6800 nebo 8080) má instrukce, které mohou vykonávat jakýkoli z následujících úkolů:

- přesouvat data a adresy mezi registry,
- přeměňovat a rotovat bity v datovém slově,
- vykonávat různé aritmetické a logické operace,
- větvit se podmíněně nebo nepodmíněně do jakékoli části programu nebo podprogramu,
- provádět různá logická srovnání,
- přičítat nebo ubírat obsahy adresy registru nebo paměti.

Skutečné mikroprocesory také mají zvláštní instrukce, které mohou být jedinečné pro určitou „rodinu" mikroprocesorů. Např. některé mikroprocesory mají různé instrukce pro přijímání dat z vnějších obvodů, jiné mají „vestavěnu" schopnost desítkové aritmetiky apod.

Programování skutečných mikroprocesorů může být jak zdlouhavé, tak časově nenáročné. Většina zájemců se může naučit psát jednoduché programy po krátké praxi a s určitou zkušeností v používání klávesnice (tastatury) nebo v ovládání vstupů prostřednictvím klopných spínačů. Mnoho programů mikroprocesorů bylo již publikováno v knihách a článcích; a jak postupuje čas, množství programů, které jsou k dispozici, se stále násobí.

Dále se podíváme podrobněji na řídicí sekci PIP-2. Uvidíme, že jsou instrukce dopravovány z programové paměti, dekódovány a vykonávány. Také se naučíme, jak změnit soubor instrukcí pro PIP-2 modifikováním mikroinstrukcí uchovaných v řídicí ROM.

Řídicí sekce PIP-2

Nejdůležitější a nejsložitější sekcí mikroprocesoru jsou jeho řídicí obvody, které dopravují v daném pořadí instrukce z paměti mikroprocesoru, dekódují je a pak vykonávají.

Řídicí sekce jako celek (obr. 28) má na starosti přesně synchronizovat pořadí jednotlivých operací, které dodávají instrukce, přenášet data, „postrkovat" čítače a vykonávat aritmetické operace.

Řídicí sekce odpovídá na instrukce, např. tím, že simultánně spojuje adresy paměti obsahující datové slovo, které má být vloženo (zdroj) a vstup příslušného registru (určení) do dvojsměrné sběrnice mikroprocesoru. Řídicí obvody pak vyšlou hodinový impuls do registru, aby jednak dokončil tuto operaci a jednak dopravil další instrukci.

i k chybám při měření, chybným úvahám a závěrům a kolektiv zde působí jako vzájemná kontrola – právě diskuse o správnosti nebo nesprávnosti dílčích úvah jsou obvykle hybnou silou vpřed. Při oživování využíváme těchto základních metod:

a) měření na zařízení obvykle za pomoci logické sondy a osciloskopu. Výsledky měření je nutné ihned psát, třeba na kus papíru, aby bylo možno po měření udělat nad výsledkem logickou úvahu, co ještě změřit, nebo zda jsme takový výsledek čekali či nikoli;

b) logická úvaha nad schématem a nad naměřenými údaji, v podstatě nad naměřenými jedničkami, nulami, impulsy a nebo průběhy v jednotlivých bodech zařízení. Výsledkem logické úvahy pak je buď další měření nebo úprava zařízení, která by měla mít za výsledek správnou funkci obvodu. Je samozřejmé, že se postupuje od nejjednodušších funkcí ke složitějším;

c) zjednodušovat problémy je důležitá metoda zejména u složitých zařízení. Je nutné „sáhnout do logiky" zařízení a některé logické vazby rozpojit ať už vnucením logické nuly nebo jedničky některým obvodům, nebo rozpojením spoje;

d) nápady, i když zdánlivě nesmyslné, přinášejí v té fázi oživování, kdy už nás nic rozumného nenapadá, uvolnění myšlenek a často se stane, že za našich podmínek, kdy oživujeme složitá zařízení bez dobrého vybavení měřicí technikou, nás přivedou na správnou cestu;

e) podrobná logická analýza obvodu, s úvažováním zpoždění obvodů (nejlépe vycházet přímo z naměřených průběhů na osciloskopu), nás přivede k řešení problémů časovacího rázu a k vyloučení možných hazardních impulsů. Přesně nakreslené vedení zemí, zátěží a rozdělení proudů a úbytků na vodičích nás přivede na řešení vf problémů.

Je samozřejmé, že oživování si usnadníme již při konstrukci zařízení tím, že umožníme některé sekvence krokovat, zhotovíme si přípravky nebo simulátory. Podmínkou je, aby kolektiv dohromady znal dobře celé zařízení, ať už se týká hardware nebo software nebo firmware (logiky, programů a mikroprogramů). V principu je nutné dodržet tyto zásady:

– každou úpravu si poznamenejte. Tu konečnou přeneste přesně do schématu;

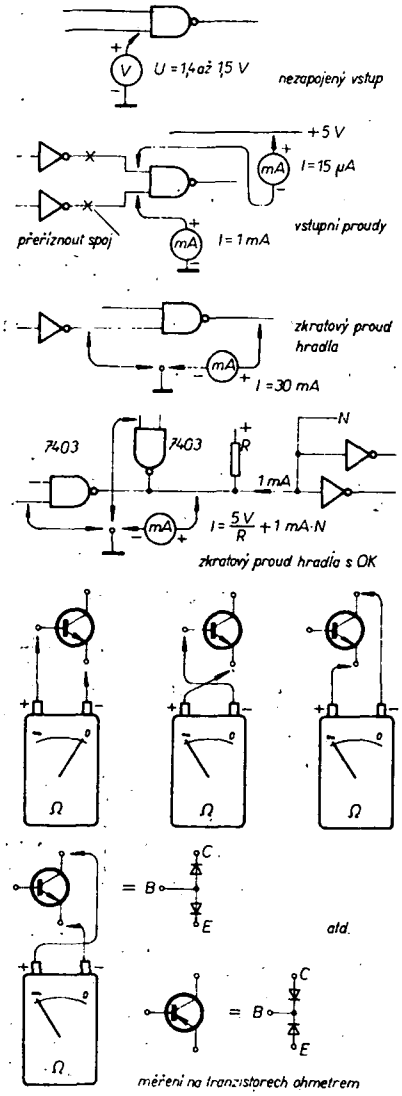
– nikdy nedělejte dvě úpravy najednou! Nemůžete pak posoudit vliv té či oné na změnu funkce;

– podaří-li se vám závadu odstranit, analyzujte všechny udělané úpravy tak, abyste přesně věděli, v čem byla chyba;

– záleželo-li při úspěšné úpravě na hodnotě součástky, rozvažte, zda bude příště stačit součástka s tou či onou tolerancí;

– nezhoršete spolehlivost celého zařízení spěšnou a neodbornou prací při realizaci vymyšlených úprav. Pájejte čistě a pro případné úpravy spojů používejte třeba červený vodič (aby bylo možno úpravy zkontrolovat). Při odstraňování vadného obvodu z desky je lepší sbroušenými štípačkami odštípat jeho vývody a pak zbytky vyndávat po jednom (u prokovovaných děr zkraťte vývody nového obvodu a připájejte ho shora). Tuzemský materiál na desky s plošnými spoji „nemá rád" odsávání a loupe se.

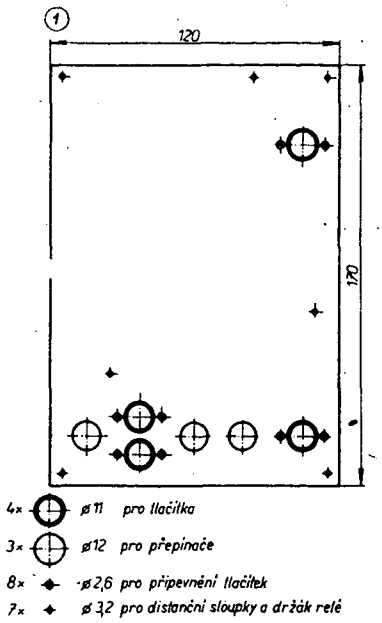
Při oživování je třeba vypracovat si vlastní metodu a neustále ji propracovávat a současně se vybavit vhodným nářadím a měřicími přípravky. Vhodné je mít logickou sondu, čítač s indikací, který by „chytil" 2 až 16 po sobě jdoucích impulsů, diody LED s odporem 1 kΩ (připájíme je do kabeláže nebo na desku na důležité výstupy), voltmetr, ohmmetr, miliampérmetr, destičku s tlačítky a generátor N impulsů a generátor signálu měnitelného kmitočtu. Dále je nutné znát základní údaje, které můžeme naměřit na vstupech a výstupech běžných logických obvodů (napětí na nezapojeném vstupu, vstupní proudy proti zemi a napájení, zkratový výstupní proud) a základní měření ohmmetrem na běžných polovodičových prvcích. Tato měření jsou znázorněna na obr. 41.



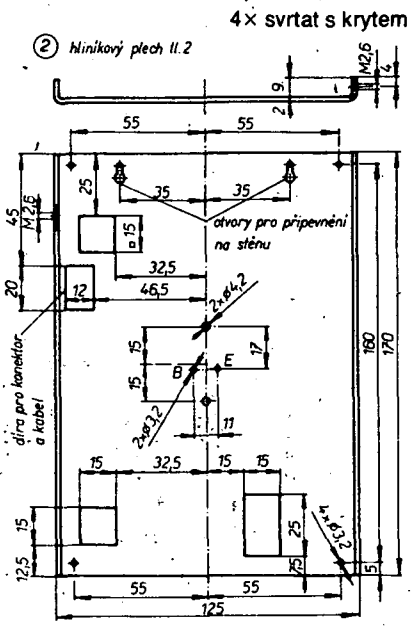
Obr. 41. Základní měření v zařízení

Zhotovení programátoru

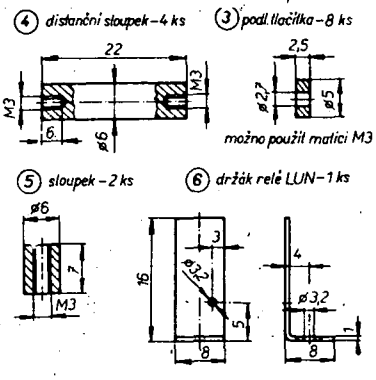
Konstrukční podklady ke zhotovení programátoru jsou na obr. 42 až 52. Desku časové základny postavíme snadno, budeme-li mít k dispozici potřebné integrované obvody a krystal. Pro ty, kteří se rozhodli šetřit nebo nemají krystal 10 MHz, je určena univerzální deska, na kterou je možno postavit děličku pro krystal s jiným kmitočtem, nebo oscilátor s NE555 s „kratší" děličkou. Při vymýšlení nového zapojení časové základny nezapomeňte na synchronizaci času pomocí přepínače AUT/RUČ! Do desky programátoru nejprve vyvrtáme všechny díry podle obr. 42. Vrtání děr velkých průměrů do kuprextitu je nesnadné a proto je lépe vyříznout díry lupenkovou pilkou. Desku osadíme součástkami a zapájíme. Na spodní stranu desky připevníme relé LUN pomocí držáku 6. Tlačítka připevníme šroubky M2,6 přes distanční podložky 3 tak, aby tlačítko procházelo hmatníkem na stranu součástek. Tlačítka musí být opatřena distančním kroužkem, který u tlačítek Isostat vymezuje hloubku stlačení, jinak by bylo možno tlačítko „promáčknout". Potom připevníme pře-



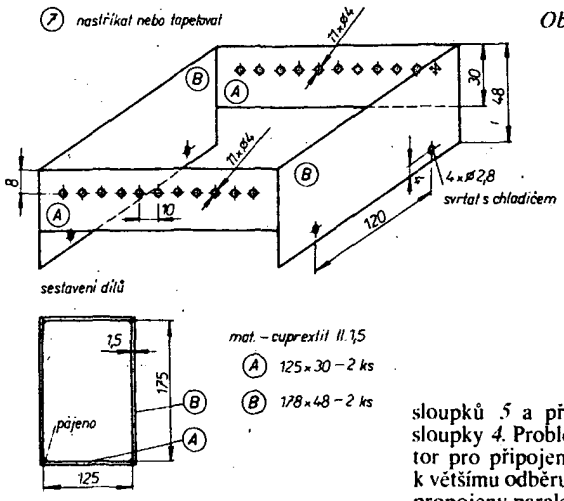
Obr. 42. Vrtací výkres desky programátoru



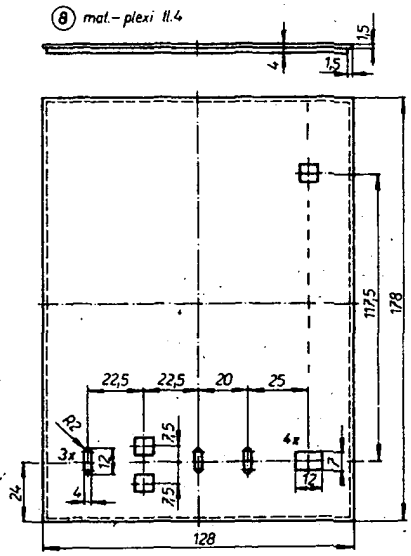
Obr. 43. Chladič



Obr. 44. Konstrukční díly



Obr. 45. Kryt



Obr. 46. Panel

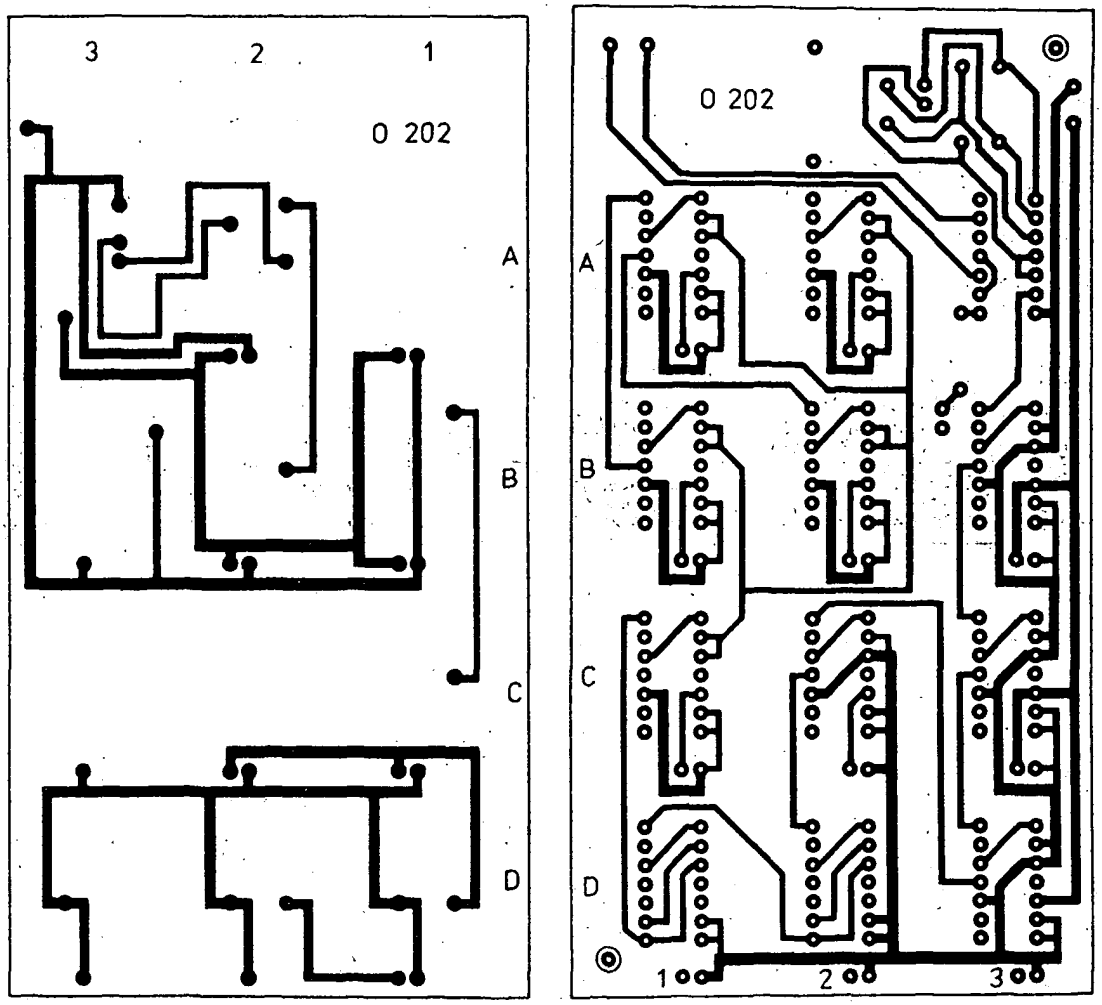
pínače a propojíme vývody relé a ovládacích prvků s ploškami na desce s plošnými spoji podle schématu.

Na chladič zhotovený podle obr. 43 připevníme pouzdro stabilizátoru a pod matice šroubů vložíme pájecí očka. Mezi vývody stabilizátoru a pájecí očka připájíme dva kondenzátory (C_4 a C_5; tyto kondenzátory musí být co nejblíže stabilizátoru).

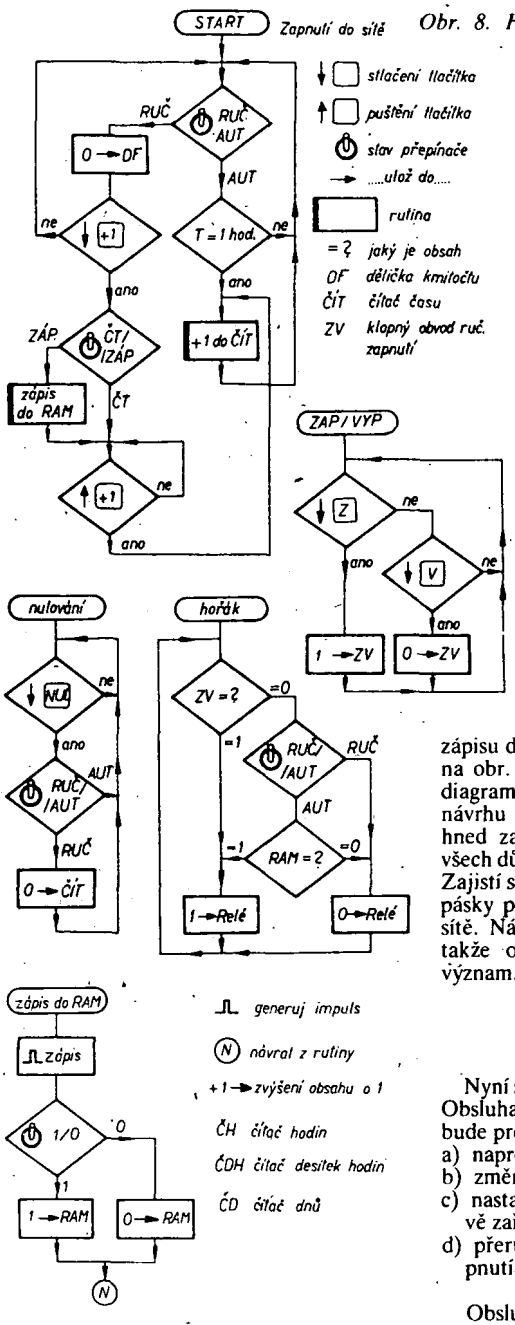
Od stabilizátoru povedou na desku programátoru tři vodiče. Propojíme napájení, hodiny a nulování mezi deskami a programátor oživíme. Potom sestavíme desky pomocí sloupků 5 a připevníme celek k chladiči sloupky 4. Problém je, jak realizovat konektor pro připojení programátoru. Vzhledem k většímu odběru proudu jsou některé špičky propojeny paralelně. Použitý konektor vznikl uříznutím z většího konektoru s roztečí kontaktů 2,5 mm a řad 5 mm na celkový počet 14 špiček. Konektor je na spodní straně desky a tělísko neleží na desce, aby bylo možno připájet kontakty. V nejhorším je možno vývody připájet přímo a konektor umístit buď do stěny nebo na kabel.

Instalace

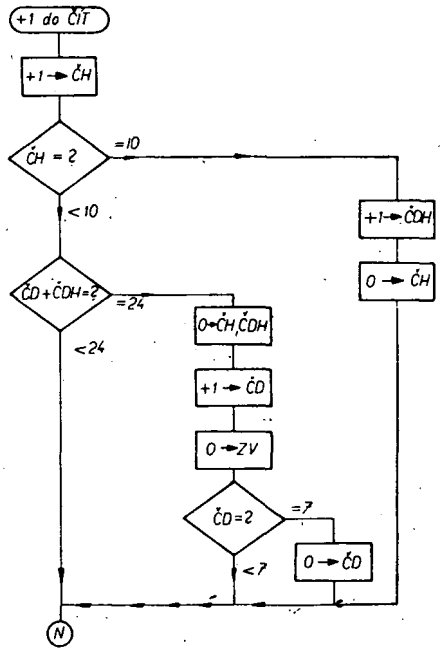
Ovládací kontakt hořáku je v podstatě zapojen do série s kontaktem termostatu, který ovládá zapnutí a vypnutí topení v závislosti na teplotě v místnosti. Mezi místem, v němž bude umístěn programátor, a kotelnou je třeba instalovat vedení minimálně o pěti vodičích. Bude-li vzdálenost větší, je dobré použít jako vodiče pro +8 V a 0 V dvě paralelně spojené žíly kabelu, aby byl úbytek na vedení při proudu 1 A co nejmenší. Pro napájení programátoru napětím +8 V je nutné postavit stabilizátor s výstupním napě-



Obr. 47. Deska s plošnými spoji O202 časové základny



Obr. 9. Vývojové diagramy rutin



Obr. 8. Hlavní vývojový diagram funkce zařízení

KROK	OPERACE	DISPLEJ	ŽÁR. PROG.
1.	přepínač RUČ/AUT do RUČ		
2.	přepínač ČT/ZÁP do ZÁP		
3.	WUL	po 00	
4.	přepínač I/O do O		
5.	+1 5x	po 05	
6.	přepínač I/O do 1		
7.	+1 2x	po 07	
8.	přepínač I/O do O		
9.	+1 8x	po 15	
10.	přepínač I/O do 1		
11.	+1 6x	po 21	
12.	přepínač I/O do O		
13	+1 3x	ú 00	
14.	+1 6x	ú 06	
	⋮		
	atd.		

svítí podle předešlého obsahu RAM. Podle nového obsahu svítí v okamžiku, kdy je stlačeno tlačítko +1

Obr. 10. Postup programování na celý týden

KROK	OPERACE	DISPLEJ	ŽÁR. PROG.
1.	přepínač ČT/ZÁP do ČT		
2.	WUL	po 00	○
3.	+1	po 01	○
4.	+1	po 02	○
5.	+1	po 03	○
6.	+1	po 04	○
7.	+1	po 05	☼
	⋮		
	atd.		

Obr. 11. Postup kontroly programu

zápisu do paměti a inkrementace čítače jsou na obr. 9. Myslím si, že uvedené vývojové diagramy nepotřebují další vysvětlení. Při návrhu jiných zařízení je vhodné zařadit hned za blok START počáteční nulování všech důležitých klopných obvodů v zařízení. Zajistí se tím např., aby se nerozběhl snímač pásky po zapnutí počítačového systému do sítě. Náš programátor bude trvale zapnut, takže obvod počátečního nulování nemá význam.

Obsluha zařízení

Nyní si stanovíme postup obsluhy zařízení. Obsluha programátoru ústředního topení bude probíhat takto:

a) naprogramování topení na celý týden,
b) změny programu,
c) nastavení správného času, třeba po opravě zařízení,
d) přerušení probíhajícího programu a zapnutí hořáku ručně.

Obsluha zařízení se dá popsat opět vývojovými diagramy. Protože však předpokládáme u našeho zařízení i obsluhu, která nezná vývojové diagramy a ani nečte AR, popíšeme obsluhu slovně s použitím grafických symbolů, způsobem používaným v návodech pro kalkulačky.

a) Naprogramování topení na celý týden

Příklad: *0. den* 0. až 5. h vypnuto, 5. až 7. h zapnuto,
7. až 15. h vypnuto,
15. až 21. h zapnuto,
21. až 24. h vypnuto,
1. den 0. až 6. h vypnuto, atd.

Postup programování pro tento případ je na obr. 10. Na obr. 11 je postup, kterým můžeme zkontrolovat, zda je naprogramování správné. Při nalezení chyby stačí přepnout na ZÁPIS, nastavit 1 nebo 0 na přepínači a stlačit tlačítko +1 a přepsat obsah RAM na správný údaj.

b) Změna programu

Program se změní, stejně jako při opravě chyby při kontrole programu.

c) Nastavení správného času

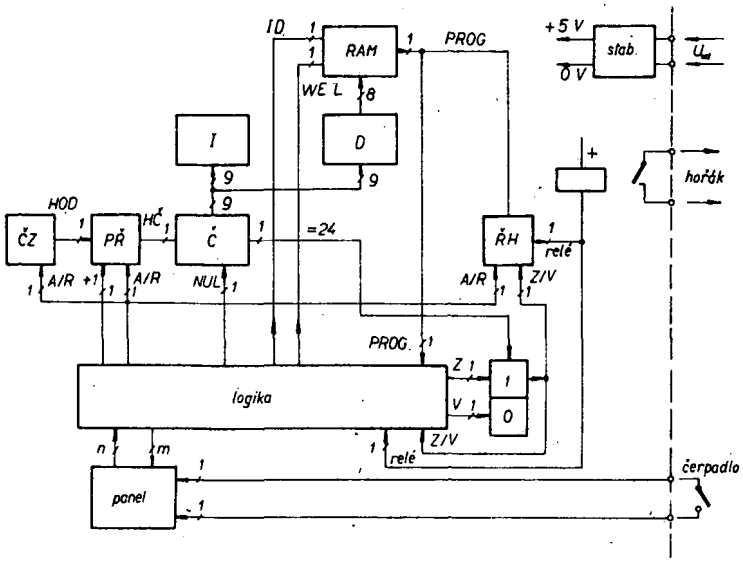
Dejme tomu, že jsme přístroj naprogramovali na celý týden a chceme spustit program, je středa 17,30 h. Přepneme na RUČNĚ a ČTENÍ a tlačítko +1 stlačíme tolikrát, až je na displeji středa 18 h. Pak počkáme na časové znamení a přesně v 18 h přepneme přepínač AUT/RUČ na AUT.

d) Přerušení probíhajícího programu a zapnutí hořáku ručně

Topení mimo program zapneme stisknutím tlačítka Z a vypneme stisknutím tlačítka V. V režimu RUČNĚ bude zapnuto topení do té doby, dokud ho nevypneme. V režimu AUT bude zapnuto topení buď do stisknutí tlačítka V, nebo, zapomeneme-li topení vypnout, až do půlnoci, pak se automaticky přepne na řízení topení podle programu. Zapnutí topení mimo program je indikováno žárovkou Ž.

Blokové schéma zařízení

V okamžiku, kdy přikročíme k sestavení blokového schématu zařízení, musíme mít alespoň částečně jasno, jaké obvody budeme



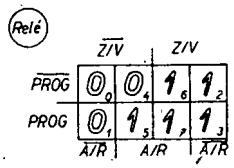
Obr. 12. Blokové schéma programátoru

mít k realizaci zařízení k dispozici a jak budou bloky realizovány. Pak teprve můžeme v blokovém schématu vyjádřit všechny potřebné vazby mezi jednotlivými bloky. V této fázi není totiž blokové schéma určeno pro znázornění funkce zařízení, jak je tomu obvykle, ale tvoří podklad pro logický návrh jednotlivých bloků. V blokovém schématu musíme proto nakreslit mezi bloky všechny signály, které budou zajišťovat, aby bloky plnily funkci, kterou jsme popsali vývojovými diagramy na obr. 8 a 9. Bloky si označíme pracovními názvy a rovněž signály označíme číslem udávajícím jejich počet, případně vhodnými názvy vyjadřujícími funkci signálu. U složitějších zařízení můžeme postupovat tak, že si nakreslíme hrubé blokové schéma zařízení a pak teprve bloková schémata jednotlivých částí. Rozdělení na zařízení na jednotlivé bloky je samozřejmě úměrné tomu, jak velký celek jsme schopni odděleně navrhnout. Jak uvidíme na příkladu programátoru, mohou být jednotlivé bloky v konečném řešení realizovány mnoha obvody nebo i pouze několika hradly. Důležité je, aby signály vstupující a vystupující z bloků dávaly návrháři představu o možné realizaci bloku. Jednotlivé bloky pak mohou být pro stejné řešení zařízení různé v závislosti na tom, jaké má návrhář zkušenosti, prakticky ověřená zapojení z jiných zařízení nebo literatury, atd. Blokové schéma programátoru je na obr. 12 a popíši jej tak, jak jsem jej sestavoval.

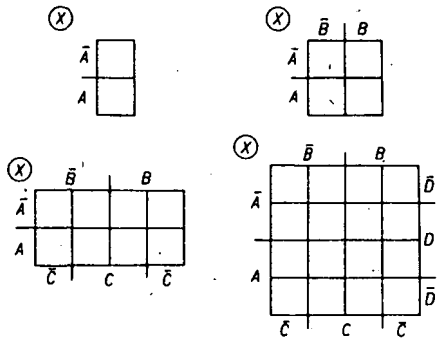
Základním blokem je časová základna ČZ (krystalový oscilátor a děličky). Výstupní signál má kmitočet o periodě 60 minut. Tento signál je označen zkratkou HOD. Jako vstupní signál u tohoto bloku je pouze signál A/R, odvozený od přepínače AUT/RUČ, který slouží k nulování děličů v režimu RUČNĚ za účelem synchronizace času se skutečným časem. Dalším blokem je přepínač PŘ, do kterého vstupují signály HOD a +1. Pomocí dalšího vstupního signálu A/R přepíná přepínač na vstup čítače hodiny HČ, buď z časové základny v režimu AUT, nebo signál odvozený od tlačítka +1 v režimu RUČNĚ. Bloky Č – čítač, I – indikace, D – dekodér a RAM – paměť jsou převzaty z obr. 4. Vstupní signály do čítače jsou „hodiny" a „nulování". Výstup čítače jde na „indikaci" a přes dekodér na adresové vstupy paměti RAM. Paměť RAM (typ 74S201) potřebuje ke své činnosti impuls pro zápis WE L a vstupní data ID. Výstup z paměti, PROG, určuje, zda má být hořák podle

Č	A/R	Z/V	PROG	Relé	Komentář
0.	0	0	0	0	RUČ: vypnuto
1.	0	0	1	0	RUČ: v RUČ nemá program vliv
2.	0	1	0	1	RUČ: zapnuto tlačítkem Z
3.	0	1	1	1	RUČ: zapnuto tlačítkem Z
4.	1	0	0	0	AUT: vypnuto
5.	1	0	1	1	AUT: zapnuto z programu
6.	1	1	0	1	AUT: zapnuto tlačítkem Z
7.	1	1	1	1	AUT: zapnuto jak tlačítkem Z, tak z programu

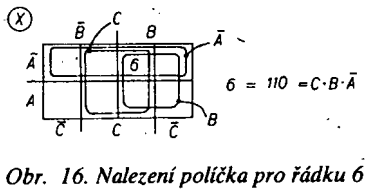
Obr. 13. Pravdivostní tabulka bloku ŘH



Obr. 14. Karnaughova mapa bloku ŘH



Obr. 15. Karnaughovy mapy



Obr. 16. Nalezení políčka pro řádku 6

programu zapnut a stav tohoto signálu je také indikován na panelu zařízení. Blok hořáku určuje, mají-li být kontakty relé Re sepnuty nebo rozpojeny v závislosti na stavu klopného obvodu ZV, výstupu RAM a signálu A/R. Klopný obvod je nastavován a nulován signály Z a V odvozenými z tlačítek na panelu a je také nulován „půlnočním" signálem (= 24). Stav klopného obvodu je na panelu indikován žárovkou Z. Blok „logika" zahrnuje obvody pro úpravu signálů z tlačítek a přepínačů a obvody zajišťující správné vazby prakticky všech signálů v zařízení. Do tohoto bloku je možné v této fázi „schovat" zbylé části zařízení, které nejsou ještě dost jasné. Dalšími bloky jsou přední panel a stabilizátor. Vezmeme-li programátor jako černou skříňku, pak jejími vstupy jsou pouze dva vodiče napájení, dva vodiče, které přenášejí na panel informaci o běhu čerpadla a dva vodiče, které přenášejí z programátoru informaci, zda má být hořák zapnut nebo vypnut. Zde jsem udělal výjimku a místo o signálech, které jsou prakticky tři, jsem začal o vodičích a to proto, aby již v této fázi bylo jasné, jak bude vypadat kabel, jaké budou problémy s rušením jednotlivých signálů a jak bude začleněn programátor jako celek do systému řízení topení.

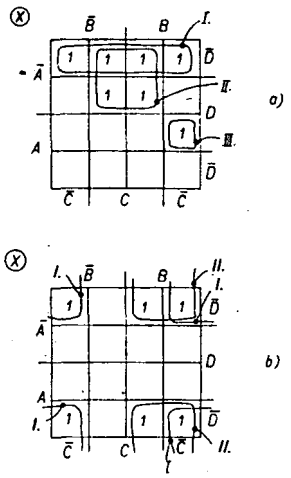
Logický návrh

Logický návrh jednotlivých bloků zařízení dělám převážně pomocí Karnaughových map. Mapa je grafickým znázorněním pravdivostní tabulky logického obvodu. Použití mapy si vysvětlíme na návrhu bloku ŘH – řízení hořáků. Nejprve si popíšeme funkci bloku ŘH slovně. Blok má dva vstupy PROG a Z/V a řídicí vstup A/R. Je-li na řídicím vstupu A/R log. 0, je zvolen režim ručně a výstup bloku „relé" sleduje vstup Z/V, tzn., že hořák je řízen pouze stavem klopného obvodu Z/V. Je-li na A/R log. 1, je zvolen režim AUT a výstup Re sleduje výstup paměti PROG nebo výstup klopného obvodu Z/V. Hořák je zapnut, je-li v paměti jednička, nebo je-li nastaven klopný obvod na zapnuto.

Nyní si zapíšeme funkci bloku do pravdivostní tabulky. Pravdivostní tabulka je na obr. 13. V prvním sloupci tabulky je dekadické vyjádření vstupní kombinace. V dalších třech sloupcích jsou všechny možné kombinace vstupních signálů a v pátém sloupci požadovaný výstupní signál při každé vstupní kombinaci; v posledním sloupci je komentář, aby bylo jasné, jak byla pravdivostní tabulka sestavena. Nyní si můžeme přepsat tuto tabulku do mapy. Každému políčku mapy odpovídá jeden řádek pravdivostní tabulky a pro snažší pochopení je číslo vepsáno do každého políčka. Do políčka mapy napíšeme požadovaný výstupní signál pro daný řádek. Mapa je na obr. 14. Vlevo nahoře v kroužku je označení výstupního signálu, pro který je mapa nakreslena. Označení signálů s pruhem znamená invertovaný signál (např. PROG NON).

Mapu je možno nakreslit pohodlně pro 1 až 4 vstupní signály. Příklady těchto map jsou na obr. 15.

(Je dobré se naučit tyto mapy kreslit rychle a hlavně si je jednotně označovat, abychom je mohli pohodlně vyplňovat. Proto je třeba dodržet pravidlo, že levé horní políčko odpovídá vždy řádku 0 v tabulce. Požadované výstupní proměnné pak napíšeme do příslušných políček. Políčko pro každou řádku leží v průsečíku řádků a sloupců, jejichž proměnné určuje příslušný řádek. Obvyklá forma zápisu je CBA, kde nejnižší bit A má váhu 1 a je vpravo. Např. řádek 6 = 110 = CBA, příslušné políčko je v mapě pro tři proměnné v průsečíku horního řádku Ā, pravé poloviny mapy B a středu mapy C, jak je to znázorněno na obr. 16. Máme-li mapu vyplněnu, můžeme s ní pracovat. Mapa je velmi dobrá k realizaci logických obvodů pomocí hradel AND a AND-OR-INVERT, které jsou u nás vyráběny. Prvním krokem je minimalizace: spojíme 2^n = 1,2,4,8, . . . sousedních políček,



Obr. 17. Minimalizace map sdružováním políček

které obsahují jedničky, do většího políčka. Čím větší výsledná políčka budou, tím lépe, takže některé jedničky zahrneme i do několika sdružených políček. Příklad je na obr. 17a. Na obr. 17b je znázorněno, že krajní políčka vlastně sousedí s políčky na opačné straně, jako by mapa byla nakreslena na kouli. Na obr. 17b jsou dvě výsledná políčka. Jedno obsahuje čtyři rohové jedničky a druhé políčko obsahuje dvě jedničky vpravo nahoře a dvě jedničky vpravo dole. Jsou-li v mapě políčka sdružena, napíšeme logickou rovnici, v níž na pravé straně napíšeme (dá se říci) souřadnice, které určují polohu sdružených políček. Takto napsané souřadnice oddělíme znaménkem + , tzn., nebo. Pro mapu a obr. 17a bude rovnice vypadat takto:

$$X = \overline{A} \cdot \overline{D} + \overline{A} \cdot C + A \cdot B \cdot \overline{C} \cdot D$$

První částí jsou čtyři jedničky v horním řádku, určené tím, že jejich poloha je v průsečíku ploch $\overline{A} \cdot \overline{D}$. Druhou částí jsou čtyři jedničky nahoře uprostřed – nahoře vlastně znamená v poli $\overline{A}$ a uprostřed v poli C. A třetí část je izolovaná jednička, kterou musíme popsat čtyřmi souřadnicemi. Rovnice, kterou napíšeme podle mapy, umožňuje přímo realizovat funkci pomocí hradel AND, OR a NOT. Pro realizaci pomocí hradel NAND použijeme na úpravu rovnice DeMorganovo pravidlo, které je na obr. 18a

$$\overline{A + B + C + \ldots} = \overline{A} \cdot \overline{B} \cdot \overline{C} \ldots \qquad a)$$

$$\overline{A \cdot B \cdot C \cdot \ldots} \quad \overline{A} + \overline{B} + \overline{C} + \ldots$$

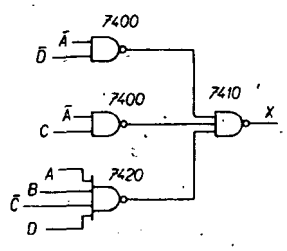
$$X = \overline{A} \cdot \overline{D} + \overline{A} \cdot C + \overline{A} \cdot \overline{B} \cdot \overline{C} \cdot D =$$

$$= \overline{\overline{A} \cdot \overline{D} + \overline{A} \cdot C + A \cdot B \cdot \overline{C} \cdot D} = \qquad b)$$

$$= \overline{\overline{A} \cdot \overline{D}} \cdot \overline{\overline{A} \cdot C} \cdot \overline{A \cdot B \cdot \overline{C} \cdot D}$$

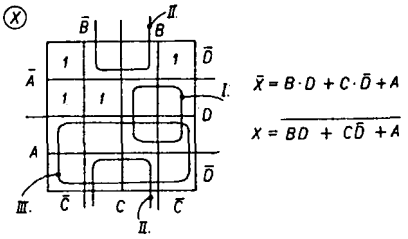
Obr. 18. DeMorganovo pravidlo

obecně a na obr. 18b pro naši rovnici s tím, že jsme pravou stranu rovnice invertovali dvakrát, což nemá vliv na logickou funkci nebo proměnnou. Podíváme-li se nyní na upravenou rovnici, můžeme ji přímo realizovat hradly NAND. Realizace je na obr. 19.

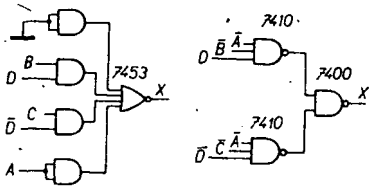


Obr. 19. Realizace funkce podle mapy

Někdy je výhodnější realizovat funkci pomocí hradel AND-OR-INVERT, které se u nás také vyráběji. Pro tuto realizaci je vhodné invertovat celou mapu, tzn. místo nul uvažovat jedničky a opačně, čímž získáme



Obr. 20. Použití inverze mapy



Obr. 21. Realizace funkce podle inverzní a běžné mapy z obr. 20

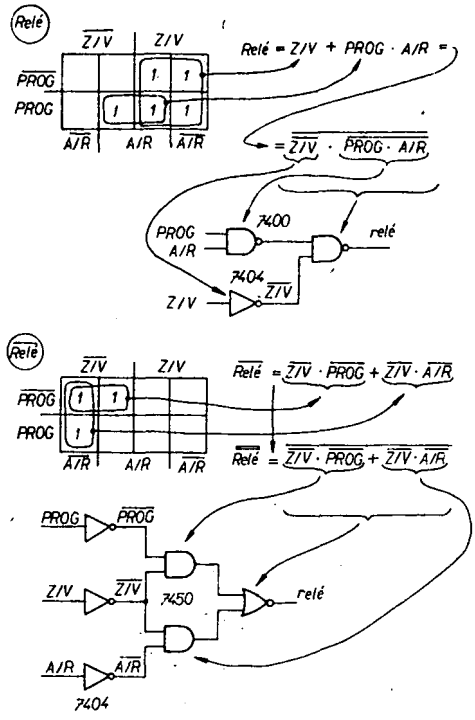
mapu pro $\overline{X}$; příklad je na obr. 20. Z mapy získáme rovnici a když invertujeme obě strany rovnice, získáme přímo rovnici pro hradlo AND-OR-INVERT. Na obr. 21 je realizace mapy z obr. 20 oběma způsoby.)

Nyní již můžeme realizovat mapu z obr. 14, která popisuje funkci bloku Řízení hořáku. Postup realizace je na obr. 22. Realizace je navržena pro obvody typu 7400 nebo 7450, protože v této fázi ještě nevíme, zda bude vhodnější použít tyto nebo jiné obvody. Obecně je možno říci, že naše realizace bloku ŘH s hradly NAND má zpoždění signálu přes dvě hradla a druhá realizace přes jedno hradlo. V našem případě to ovšem nehraje roli a tak si počkáme, zda po návrhu všech obvodů nebude náhodou přebývat volná polovina obvodu 7400 nebo 7450. Invertory na obr. 22 jsou kresleny pouze pro úplnost návrhu. Při konečném spojování bloků bude možno použít místo výstupu Q klopného obvodu Z/V výstup $\overline{Q}$ a signál A/R bude mít také jistě přímou i invertovanou hodnotu, protože jde z přepínače.

Realizace časové základny

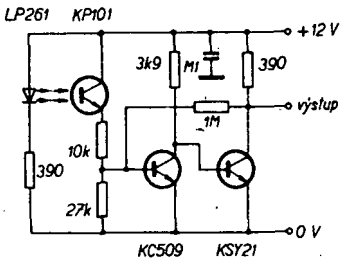
Blok časové základny je po stránce logického návrhu nejjednodušším blokem programátoru. Návrh je závislý na kmitočtu použitého krystalu a děličky kmitočtu až na periodu 1 h. Časovou základnu jsem navrhl jako zvláštní stavební díl, aby bylo možno použít i krystal jiného kmitočtu. Pro ty, kteří nebudou mít k dispozici tolik obvodů 7490 nebo 7493, bych měl pouze několik nápadů. Kdo již má doma postavené nebo koupené digitální hodiny, může si buď programátor přistavět přímo k hodinám přidáním čítače dnů, ovládacích prvků a paměti, nebo z nich vyvést na konektor impulsy s periodou jedné hodiny a ty použít místo výstupu časové základny. Další možností je použít obvod NE555 a udělat z něj časovač s periodou, kterou by bylo možno vydělit levněji (AR B2/79, str. 56 a 59), nebo přímo s periodou 1 h. Přesnost hodinových impulsů pro programátor nemusí být velká, neboť na čtvrthodině by při týdenním programu nemělo záležet a časovač by bylo možno kdykoli zasynchronizovat na celé hodiny přepnutím přepínače RUČ/AUT.

Poslední nápad je ryze amatérský a můžete se pokusit o jeho realizaci. Máte-li doma elektrické nástěnné hodiny nebo budík, pokuste se sejmout polohu minutové ručičky tak, aby snímač dal impuls při každém průchodu ručičky dvanáctkou. Jako nej-



Obr. 22. Realizace mapy z obr. 14 (mapa bloku ŘH)

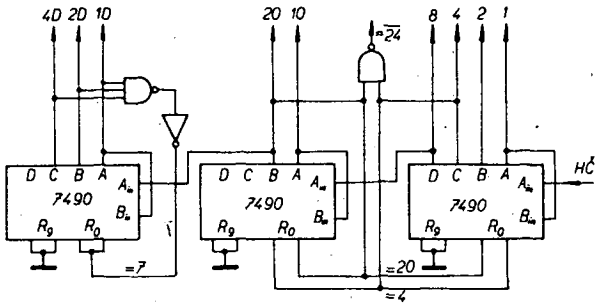
vhodnější pro podobné aplikace je snímač fotoelektrický. Ručička přeruší světelný tok dopadající ze zdroje světla na fotocitlivý prvek a signál se zesílí na úroveň logiky TTL. Zesilovač v takovém snímači musí mít zavedenu hysterezi pomocí zpětné vazby z výstupu, aby nedošlo ke kmitání výstupního signálu, je-li ručička právě na rozhraní citlivosti snímače. Jako zdroj světla je nejvhodnější dioda LED, vyzařující infračervené paprsky, a jako snímací prvek fototranzistor. Příklad zapojení takového snímače je na obr. 23.



Obr. 23. Příklad zapojení fotoelektrického snímače polohy

Realizace čítače

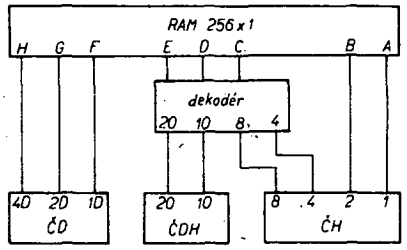
Blokové schéma čítače je na obr. 3 a vývojový diagram funkce bloku čítače na obr. 9. Z vývojového diagramu je patrno, že funkci čítače hodin ČH plní obvod 7490. Jako čítače desítek hodin použijeme polovinu dalšího obvodu 7490 a abychom splnili vývojový diagram, vydekódujeme z čítače hodin a desítek hodin stav 24 a výstupem dekodéru vynulujeme čítače hodin a klopný obvod ZV. Jako čítač dnů použijeme opět obvod 7490, jemuž vydekódujeme stav 7 a výstupem dekodéru čítač vynulujeme. Realizace čítače je na obr. 24. Čítač je nakreslen opačně, než je zvykem, a to proto, aby nejnižší bit byl



Obr. 24. Realizace čítače hodin a dnů

Dny				Des. hodin			Hodiny				
4D	2D	1D	Pozn.	20	10	Pozn.	8	4	2	1	Pozn.
0	0	0	Po	0	0	0	0	0	0	0	
0	0	1	Ú	0	1	1	0	0	0	1	
0	1	0	St	1	0	2	0	0	1	0	
0	1	1	Č	1	1	nepouž.	0	0	1	1	
1	0	0	Pá				0	1	0	0	
1	0	1	So				0	1	0	1	
1	1	0	N				0	1	1	0	nepoužito,
1	1	1	nepouž.				0	1	1	1	jsou-li des.
							1	0	0	0	hodin = 2
							1	0	0	1	
							1	0	1	0	nepoužito
											vůbec
							1	1	1	1	

Obr. 25. Zjednodušená pravdivostní tabulka vstupů dekodéru

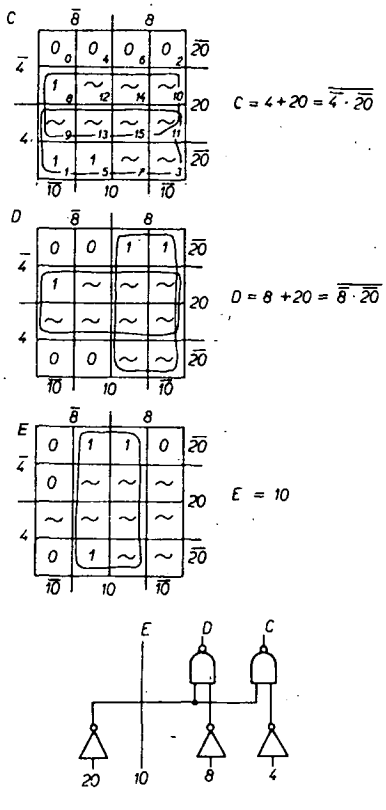


Obr. 26. Zjednodušený problém dekodéru

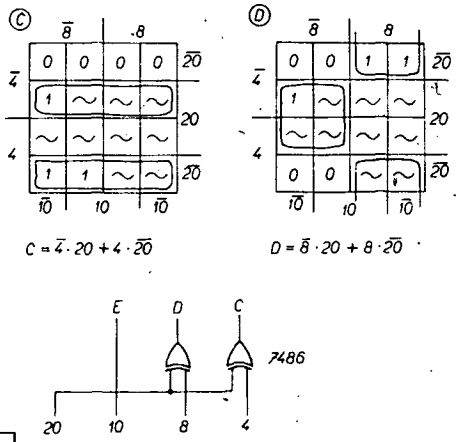
Ř	20	10	8	4		E	D	C
0	0	0	0	0		0	0	0
1	0	0	0	1		0	0	1
2	0	0	1	0		0	1	0
3	0	0	1	1		~	~	~
4	0	1	0	0		1	0	0
5	0	1	0	1		1	0	1
6	0	1	1	0		1	1	0
7	0	1	1	1		~	~	~
8	1	0	0	0		0	1	1
9	1	0	0	1		~	~	~
10	1	0	1	0		~	~	~
11	1	0	1	1		~	~	~
12	1	1	0	0		~	~	~
13	1	1	0	1		~	~	~
14	1	1	1	0		~	~	~
15	1	1	1	1		~	~	~

Obr. 27. Pravdivostní tabulka dekodéru z obr. 26

vpravo tak, jak jsme zvyklí psát binární čísla. Z obrázku je vidět, že pro realizaci funkce čítače jsme obsadili nulovací vstupy čítačů a musíme proto ještě logickými obvody vyřešit nulování z tlačítka NUL. Tak jednoduché logické obvody řešíme většinou přímo



Obr. 28. Realizace dekodéru pomocí map



Obr. 29. Realizace dekodéru hradly exclusive-or

z hlavy a přemýšlení máme zjednodušeno tím, že TESLA nevyrábí jiné obvody než NAND a musíme se smířit s tím, že naše zařízení mají plno zbytečných invertorů. Čítač s nulováním je uveden v konečném zapojení programátoru.

Realizace dekodéru

Dekodér, který by převedl 9bitové číslo na 8bitové, nelze celý řešit mapou. Proto musíme nejprve provést zjednodušující úvahy, abychom si řešení usnadnili. Při návrhu dekodéru budeme používat označení výstupů z čítače podle obr. 24. Úkolem dekodéru je vlastně ušetřit jeden bit, protože paměť RAM má pouze 8 adresových vstupů. Nejprve si napíšeme všechny možné stavy vstupů dekodéru do zjednodušené pravdivostní tabulky (obr. 25). Z tabulky je vidět, že bity pro dny používají prakticky všechny kombinace, až na jednu, a že nejnižší dva bity hodin jsou také plně využity. Nejméně jsou využity kombinace bitů 20 a 10 a bitů 8 a 4. Proto si problém návrhu dekodéru zjednodušíme tak, jak je to na obr. 26. Nyní již můžeme napsat pravdivostní tabulku dekodéru. Pravdivostní tabulka je na obr. 27. Na levé straně jsou všechny možné vstupní kombinace a na pravé jsou k nim přiřazeny hodnoty výstupních signálů. Pro ty stavy, které nemohou nastat, jsou hodnoty výstupních signálů libovolné (1 nebo 0) a v tabulce jsou označeny vlnovkou. Hodnoty výstupních signálů na pravé straně je možno přiřadit zcela libovolně, neboť nám je jedno, na kterou adresu v RAM se údaj zapíše; je však výhodné je volit tak, aby byly co nejvíce podobné levé straně. Jediné rozhodnutí při vyplňování tabulky musíme udělat v řádku 8, kam napíšeme kombinaci, která ještě nebyla a je co nejvíce podobná levé straně. Potom již můžeme pomocí map dekodér realizovat – postup je na obr. 28. Vzhledem k minimalizaci jsou sdružena políčka do maximální možné míry. Děláme-li s mapami déle, můžeme pak odhadnout, jak sdružit políčka tak, abychom mohli použít obvodové řešení, které má třeba menší zpoždění nebo využívá obvodů, které nám na desce zbudou, nebo řešení, které má méně spojů atd. Příklad jiné realizace téhož dekodéru je na obr. 29. Jednoduchost dekodéru dokazuje, že naše úvahy nad obr. 4 byly správné a že řešení B je jednoduché. Řešení podle obr. 4c by vyžadovalo dva binární čítače 7493.

Realizace dalších bloků

Ostatní bloky programátoru navrhujeme buď obdobně, nebo ty, které jsou jasné, převezmeme z literatury nebo z vlastní Kuchařky, do které sbíráme osvědčená a vy-

zkoušená zapojení. Při návrhu obvodů interface (neboli styku) dbáme na to, aby vstupní i výstupní signály logiky byly správně ošetřeny. Vstupy a výstupy se ošetřují proto, aby se jednak zamezilo pronikání rušení z cizích zdrojů rušení do našeho systému a jednak chránily vstupy a výstupy před zničením. Používáme rad, které dávají výrobci integrovaných obvodů a neustále se učíme z literatury.

Celkové schéma

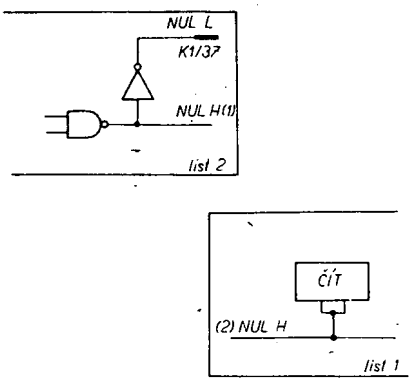
Celkové schéma zařízení by mělo být přehledné a především by mělo stačit k tomu, abychom se my, případně i někdo jiný vyznal v zařízení, má-li k dispozici schéma. Někteří naši výrobci se postupnou normalizací a racionalizací dostali až do stadia, kdy jsou jejich schémata naprosto nesrozumitelná. Z vlastních zkušeností mohu potvrdit, že dokumentace většiny zahraničních zařízení je tak srozumitelná, že stačí schéma a několik poznámek, výpisy obsahů paměti ROM a funkce zařízení je naprosto jasná. Úroveň dokumentace se pak odráží v době, za níž lze nalézt chybu a závadu opravit, tj. v rychlosti servisu. Pro ty, kteří kreslí častěji schémata větších zařízení (třeba multimetr), několik rad:

a) rozdělíme schéma na několik listů. Složité schéma na jednom listu je nepřehledné a má více spojů než značek obvodů;

b) důležité signály v zařízení označte názvy, které zkratkou vyjadřují jejich funkci a za název napište L nebo H podle toho, zda je aktivní signál o úrovni log. 0 nebo 1. Například čítač nulujeme jedničkou – signál bude proto označen NUL H. Obvod 7474 nulujeme nulou – signál bude označen RES L;

c) signály, které jsou vnitřní (nejsou vyvedeny na konektor), avšak přicházejí z jednoho listu schématu na druhý, označte na listu, odkud vycházejí, zkratkou, aktivní úrovní a číslem listu, na němž je ta část zapojení, do níž se signál přivádí, a opačně, na listu, kde je zapojení, do něhož signál vchází, číslem listu, odkud signál „vyšel", zkratkou a aktivní úrovní.

Příklad je na obr. 30. Tento způsob se používá i v rámci jednoho listu, aby se



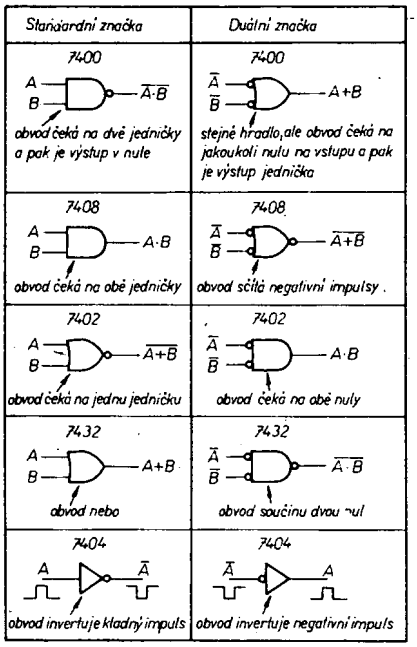
Obr. 30. Značení signálu mezi jednotlivými listy schématu a

nemuselo kreslit třeba 8 linek sběrnice procesoru přes celé schéma tam a zpátky. K signálům, které vedou na konektor, napište název, číslo konektoru a špičky;

d) k důležitým obvodům napište zkratkou jejich funkci. Například registr instrukce IR, klopný obvod přerušení INT (Interrupt = přerušení). atd. Nebojte se používat zkratek z angličtiny, neboť je to mateřský jazyk výpočetní techniky a lépe se pak vyznáte v literatuře. Vhodné je i označit celé bloky názvem funkce přímo do schématu;

e) zvlášť pak napište seznam signálů na konektorech, kabeláž mezi deskami, případně i seznam signálů mezi jednotlivými listy. Tyto seznamy jsou důležité pro kontrolu návrhu desek s plošnými spoji i pro měření.

(Uvedené rady jsou vlastně souhrnem poznatků ze studia dokumentace zahraničních zařízení z oblasti výpočetní techniky. V této speciální literatuře, která mnohdy leží bez povšimnutí ve skříni u majitelů zahraničních zařízení a čeká statečně, až se zařízení poškodí, je více poučení, než v obvyklé technické literatuře. V dokumentaci jsou uvedeny obvykle i změny, které ukazují, jak se postupně inženýři dostávali ke konečnému stavu zařízení. Zajímavé je i studovat vývoj v oboru podle dokumentací zařízení od stejné firmy, ale patřící výrobkům z různých let. Vidíme snahu zvětšit „chytrost", spolehlivost a výkonnost zařízení a snížit jeho cenu, cenu montáže, zjednodušit měření a servis zařízení. Málokdy vidíme v takových zařízeních složité obvody, zavánějící touhou po patentu, jako u nás. Na druhé straně můžeme pozorovat, jak konzervativní jsou některé firmy, jak se drží osvědčených součástek a zapojení, dokud je konkurence nepřinutí k radikální inovaci zařízení. Za dokumentací každého zařízení vidíme však nesmírnou a cílevědomou práci mnoha lidí, kteří se na jeho vývoji podíleli – a z té bychom si měli vzít příklad. U nás je trochu problém v tom, že většina konstruktérů je zkažená strojařským pohledem na věc, neboť strojařina má u nás větší tradici a je preferovanější než elektronika, a tak není zavedeno psát do schémat poznámky, pravdivostní tabulky obvodů nebo používat duální značení (obr. 31).



Obr. 31. Duální značení logických obvodů (používané v zahraničí). I ve složité logické síti lze ihned poznat, jakou funkci hradlo plní. Používá se i při evropském značení (FACIT)

Většina výrazů se překládá do češtiny, čímž se zabývá mnoho normalizačních komisí a mezitím než vyjdou normy, se výraz třeba přestane ve světě používat, protože vývoj v eletronice letí moc rychle. Zde bychom si měli vzít příklad od sovětských kolegů, kteří používají anglické výrazy tak, jak je slyšeli a zabývají se o to intenzívněji jejich fyzickou realizací. Proto bych chtěl poradit vám, mladým, berte věci tak, jak jsou. Nikdy si o nějakém zařízení nemyslete, že je moc dobré nebo moc špatné, prostudujte dokumentaci, rozeberte ji do co největších detailů a poučte se. Možná, že při prvním prohlížení zjistíte, že nerozumíte ničemu, to je normální. Nevzdávejte se, čtěte a studujte znovu a další a další dokumentaci a ono to půjde. Na své cestě za dobrodružstvím zvané elektronika se setkáte s mnoha problémy, pramenícími z nepochopení a z nechuti riskovat od druhých nebo z vlastní neznalosti a podceňování rad zkušenějších; jednou však zjistíte, že už něco umíte a pak se vám najednou za dokumentací zařízení objeví člověk, inženýr, který měl rád odpory 1 kΩ a hradla NOR a proto jich je na schématu více než na schématu jiné desky téhož zařízení, kterou konstruoval asi jeho kamarád, který měl rád odpory 4,7 kΩ a hradla 7400. Pak teprve budete umět číst základní poselství elektroniků své době, schémata.)

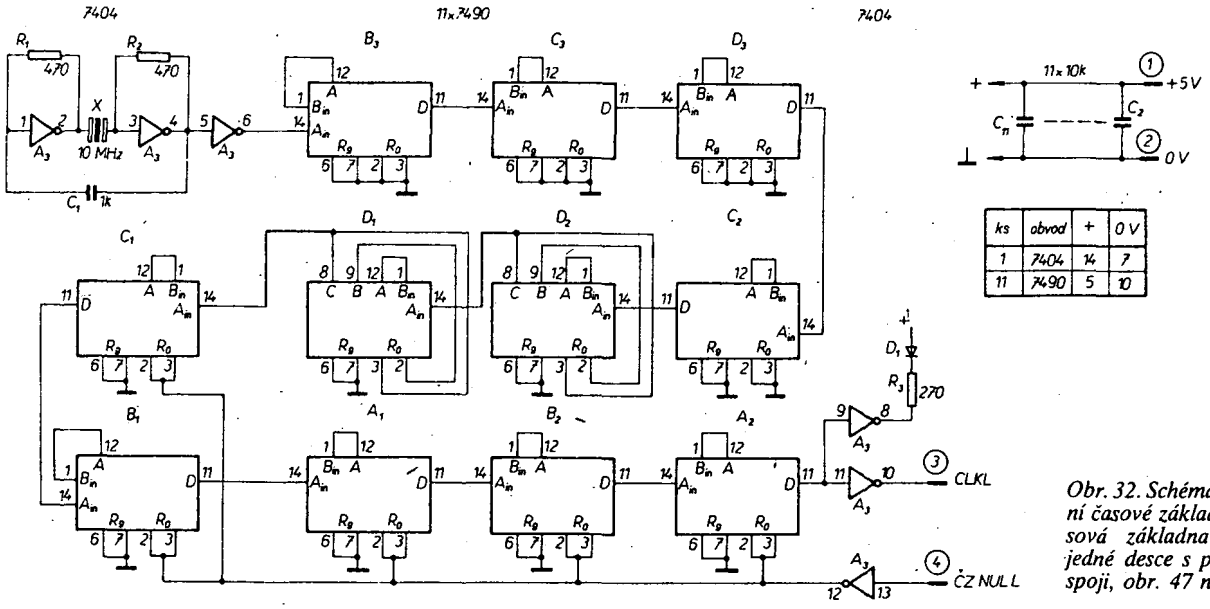
Celkové schéma našeho zařízení je na obr. 32 – deska časové základny a na obr. 33 – deska programátoru. Deska časové základny nevyžaduje prakticky vysvětlení. Pouze děliče šesti bylo nutné zařadit do řetězce tak, aby nevadilo, že mají obsazeny nulovací vstupy. Synchronizace pak nuluje pouze část děličů, aby nebyla překročena povolená zátěž invertoru. Výstup z posledního čítače je oddělen na výstupu z desky invertorem, protože není vhodné vést výstupní signál přímo do kabeláže, což platí pro klopné obvody obecně. Zbytek invertoru je pak použit pro oddělení nulování, což by nemuselo být a pro indikaci výstupního signálu diodou LED, která je potřebná pouze pro oživování a kontrolu funkce desky pomocí hodinek nebo časového znamení. Deska programátoru vychází z navržených obvodů. Při kreslení celkového schématu spojujeme předem navržené bloky a může se stát, že nás napadne, jak ušetřit dekodér stavu 7 v čítači dnů pomocí výstupu 7dekodéru pro indikaci dnů. Takový nápad prověříme a zahrneme ho do schématu. Je samozřejmé, že v této fázi ještě do schématu neoznačujeme pozici obvodu na desce a číslo vývodu u obvodů, kde to není jednoznačné. Tyto údaje doplníme až v etapě konstrukce. Naopak již před konstrukcí bychom měli rozdělit obvody zařízení na jednotlivé desky, abychom schémata nemuseli překreslovat. Ty obvody, které jsme nenavrhovali v předešlém výkladu, jsem buď spočítal z hlavy, nebo jsem je převzal z katalogů a příruček výrobců obvodů a literatury, nebo ze své Kuchařky. Tím jsem skončil etapu logického návrhu a až do etapy oživování si vždy myslím, že mám vše správně.

Konstrukce

Ke konstrukci zařízení potřebujeme schéma, obrázek ovládacího panelu, konstrukční katalog součástek, nebo lépe hromádku součástek, které přicházejí v úvahu, a posuvné měřítko, milimetrový papír, tužku, velkou mazací pryž, inspiraci, představu o instalaci a ovládání zařízení, přepočítat si výkonové zatížení součástek a výkony, které musíme „uchladit" a trochu sebezapření, abychom mohli konstruovat i se součástkami, o nichž víme, že jsou nevhodné (jiné nejsou k dispozici). Musím ovšem podotknout, že vy, amatéři, jste na tom lépe, protože kdybychom my, profesionálové, mohli použít některé ze součástek, nabízených v inzerci, dělalo by se nám mnohem lépe.

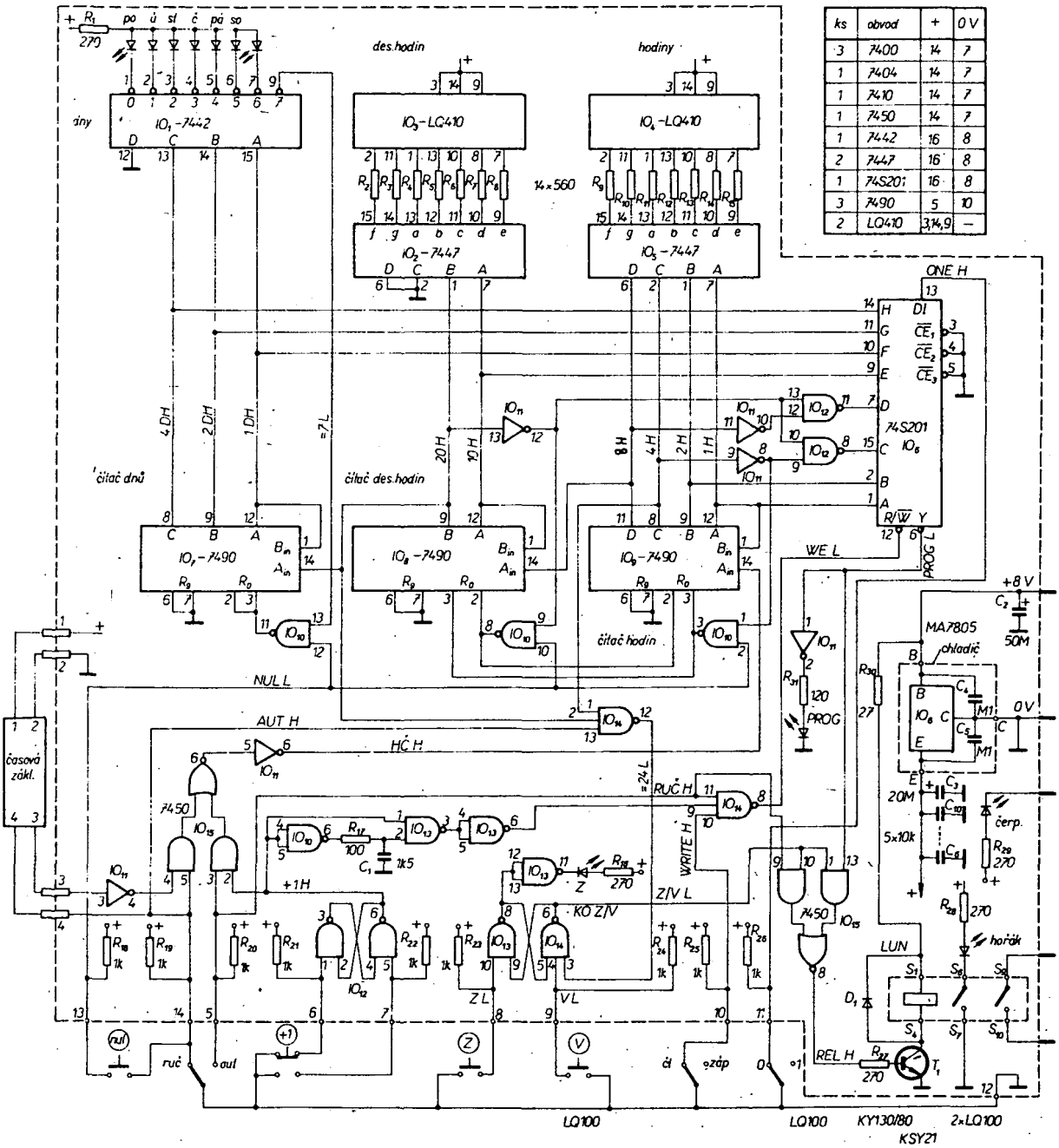
První krok je navrhnout mechanickou koncepci zařízení. Můžeme se rozhodnout pro již osvědčené řešení, které třeba zapadá rozměrově do řady zařízení, které jsme již udělali, nebo pro řešení nové. Mechanická koncepce programátoru vyšla z požadavků co nejmenšího rozměru a byla ovlivněna požadavkem řešit časovou základnu univerzálně. Proto se programátor skládá z desky programátoru, desky časové základny, ze stabilizátoru a z krytu. Ovládací prvky jsou připevněny k desce programátoru, aby odpadla kabeláž mezi panelem a logikou.

Dalším krokem je rozmístění prvků na panelu. Zde byly pouze dva výchozí požadavky, ověřené pokusy na papírovém panelu. Za prvé, aby bylo možno jednou rukou mačkat tlačítko +1 a druhou volit 1 nebo 0, a za druhé, aby nebylo možno omylem stisknout tlačítko NUL, neboť jeho nechtěným stlače-



ks	obvod	+	0 V
1	7404	14	7
11	7490	5	10

Obr. 32. Schéma zapojení časové základny (časová základna je na jedné desce s plošnými spoji, obr. 47 nebo 48)



ks	obvod	+	0 V
3	7400	14	7
1	7404	14	7
1	7410	14	7
1	7450	14	7
1	7442	16	8
2	7447	16	8
1	74S201	16	8
3	7490	5	10
2	LQ410	3;14;9	–

Obr. 33. Schéma zapojení programátoru (programátor je na jedné desce s plošnými spoji, obr. 49 a 50)

ním při programování bychom museli začít programovat opět od začátku.

Potom jsem určil velikost desek odhadem podle počtu obvodů. Protože zařízení používá bipolární integrované obvody, zbytečně rychlé (a „výkonové") pro tyto aplikace, přepočítal jsem odběr proudu zařízením ze stabilizátoru +5 V. Výsledek je na obr. 34.

Obvod	Odběr [mA]	Kusů	Celkem [mA]
7400	8	3	24
7404	8	3	24
7410	6	1	6
7450	6	1	6
7442	28	1	28
7447	64	2	128
7490	29	14	406
74S201	100	1	100
LED	10	12	120
DISPLEJ	70	2	140
odpory na +	5	4	20 / 1 A

Obr. 34. Výpočet odběru zařízení

Celkový odběr proudu bude až 1 A ze zdroje +5 V. To znamená, že při vstupním napětí 8 V bude ztrátový výkon stabilizátoru 3 W a součástek na deskách 5 W. Na tak malé zařízení je 8 W již velký ztrátový výkon. Desky i chladič musí proto být svisle (zařízení bude viset na stěně) a kryt musí umožnit proudění vzduchu i za cenu, že zařízení nebude zcela zakrytováno. Povolené oteplení součástek podle katalogu je sice značné, současně má však negativní vliv na spolehlivost zařízení. Součástky s větším ztrátovým výkonem (7447, LQ400, 74S201) umístíme proto na desce pokud možno nahoře. Ovládací prvky, přepínače a tlačítka vybereme ze dvou typů, které se u nás vyrábí. Tlačítka použijeme typu Isostat a přepínače síťové, páčkové. Páčkové přepínače jsou výhodnější než tlačítkové, protože obsluha lépe rozezná polohu, do které jsou přepnuty. Poloha tlačítkových přepínačů by musela být indikována diodami LED. Diody LED použijeme jako indikační prvky PROG., HOŘÁK, ČERP., Z a pro indikaci dne. Jednotlivé díly zařízení budou propojeny přímo, bez konektorů, které jsou velmi drahé. Výstup ze zařízení bude přes konektor, tam je to nutné.

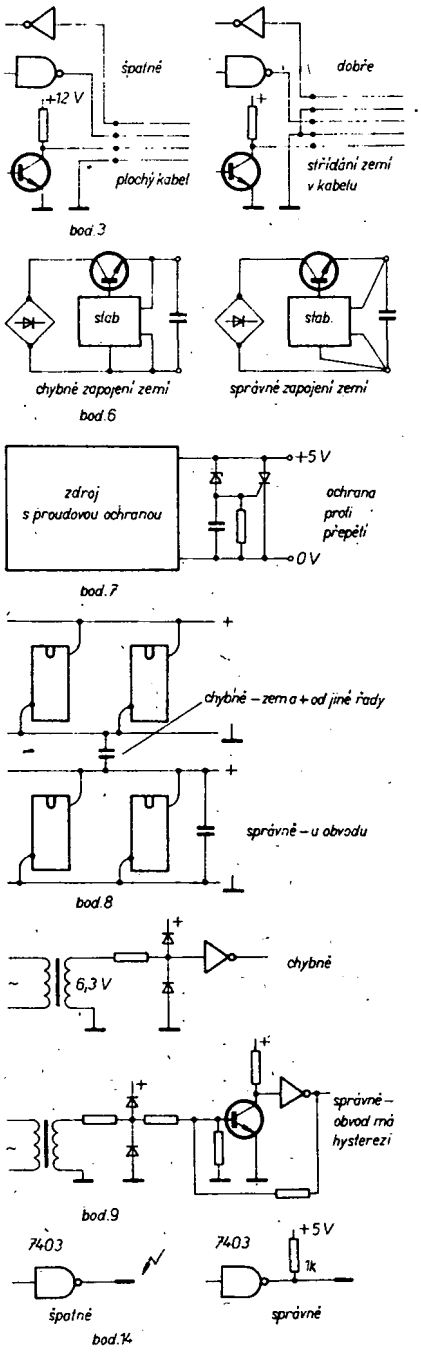
Opravu zařízení si usnadníme nezávislostí vnitřku na krytu. Panel zařízení bude z červeného organického skla, které působí jako filtr pro diody LED a necháme-li ho vcelku, nemusíme tak přesně osazovat desku diodami LED, jako kdybychom měli v panelu pouze okénka z organického skla.

Tím by byly konstrukční úvahy skončeny a zopakujme si proto základní hlediska na každou konstrukci: spolehlivost, výrobní cena, opravitelnost zařízení a obsluhovatelnost zařízení. Celkový vzhled (design) zařízení v tomto výčtu chybí, neboť je převážně výsledkem snahy o splnění uvedených hlavních cílů. Lisování krytů zařízení z plastických hmot nemá jako prvořadý cíl zvýšit estetičnost, ale zlevnit výrobu, estetický vzhled zařízení je z velké části dán použitým materiálem a technologií.

Kontrola

Kontrola konstrukce zařízení je postup, který nevynechá při své práci žádný vývojář. Zařízení je třeba kontrolovat systematicky, proto vám navrhnu několik bodů, které si po konstrukci projděte a promyslete si, zda máte vše správně. Některé body jsou znázorněny na obr. 35.

0. Je zařízení bezpečné? Nemůže ohrozit obsluhu nebo opraváře?



Obr. 35. Osvědčené „drobnosti" z praxe pro návrh desek s plošnými spoji a pro konstrukci zařízení

1. Mám všechno poznamenáno, abych mohl zařízení třeba za rok opravit a nebo ho postavit ještě jednou?
2. Je zajištěno chlazení součástek? Neohřívá některá součástka jinou, na jejíž teplotě záleží? Nemám v cestě proudění vzduchu nějakou přepážku, kryt nebo součástky postavené třeba na výšku?
3. Je zajištěno stínění součástek a vodičů v kabelech tam, kde na tom záleží? Je vedení signálů v kabelech promyšleno tak, aby se signály neovlivňovaly?
4. Mám konektory popsány nebo klíčovány tak, abych je nemohl někdy přehodit? Stalo by se něco, kdybych to udělal?
5. Může obsluha snadno vyměnit součástky s omezenou dobou života (žárovky, pojistky)?
6. Je dobře proveden rozvod napájecích napětí a zemí? Má rozvod minimální indukčnost a odpor? Promyslel jsem spojení zemí ve zdrojích, zemí různých zdrojů mezi sebou, spojení s kostrou a propojení zemí se spolupracujícím zařízením?
7. Je zajištěna ochrana zdrojů proti zkratu a ochrana integrovaných obvodů při zvětšení U_{cc}? Může ke zkratu nebo přepětí dojít při měření, třeba zkratováním sousedních špiček konektoru?
8. Mám zajištěno blokování na deskách u každého pouzdra sekvenčního obvodu, výkonového budiče, operačního zesilovače nebo komparátoru? Mám zajištěno blokování napájení ostatních pouzder tak, aby na jedno pouzdro „vyšel" alespoň 1 nF z keramického kondenzátoru společného pro max. 10 pouzder?
9. Mám na vstupech do logiky nebo v místech, kde může být hrana signálů horší, tvarovací obvody s hysterezí?
10. Není kapacitní zátěž výstupů obvodů větší než 0,5 až 1 nF? Mám tam, kde tranzistor nebo obvod vybíjí větší kapacitní zátěž, omezovací odpor alespoň 100 Ω?
11. Jsou ošetřeny nepoužité vstupy obvodů?
12. Jsou ošetřeny vstupy z kontaktů tak, aby nevadily odskoky kontaktů při přepínání?
13. Jsou ochráněny obvody a tranzistory, které mají indukční zátěž?
14. Mám chráněny vstupy a výstupy ze systému proti přepětí, opačné polaritě nebo proti statické elektřině? Jsou kabely z „twistu" a jsou-li delší, jsou přizpůsobeny?
15. Promyslel jsem diagnostiku zařízení? Mám možnost indikovat nějaké stavy nebo krokovat hodiny, promyslel jsem si popř. přípravek na testování?
16. Mám správné odpory na výstupech s otevřeným kolektorem? Nespojil jsem omylem výstup běžných obvodů? Mám odpory i na třístavových výstupech tam, kde obvod může zůstat ve třetím stavu?
17. A co hazardní impulsy, najdu je při oživování, nebo ještě popřemýšlím?
18. Nevedu do kabeláže (nebo ven ze systému) výstupy klopných obvodů, aniž bych je oddělil tranzistorem nebo vhodným obvodem?
19. Mám zajištěno po zapnutí zařízení správné nastavení klopných obvodů, čítačů, registrů? Rozběhne se po zapnutí vždy oscilátor? Nevzniknou v monostabilních obvodech po zapnutí signály, které by mohly něco způsobit?
20. Udělal jsem něco proti pronikání rušení ze sítě a z jiných zdrojů do zařízení? Nebudu naopak generovat rušení já, svým zařízením?
21. Nevyužívám příliš max. povoleného zatížení součástek? Nejsou někde maximální povolené parametry překročeny?

Návrh desek s plošnými spoji

Na způsob návrhu plošných spojů se názory různí. Zásadně je nutné udělat návrh v měřítku 2 : 1 a nejlepší je použít čtverečkovaný papír, který má čtverečky 5 × 5 mm a dobrý soutisk obou stran. Pak odpovídá jeho rastr právě rastru 2,5 × 2,5 mm, neboli rozteči vývodů konektorů, integrovaných obvodů a ostatních součástek. Někdo navrhuje oboustranné desky s plošnými spoji raději tak, že kreslí na jednu stranu papíru obě strany, jako kdyby viděl „skrz desku". Musí pak používat dvě barvy a barevné tužky se špatně gumují. Protože je však gumování mou nejčastější činnosti při návrhu, kreslím raději obyčejnou tužkou na obě strany papíru. Koordinaci prací na obou stranách bu-

doucí desky zajišťuji tak, že jakmile nakreslím nová kolečka na spodní stranu, překreslím je hned i na horní, buď se podívám proti světlu, nebo si polohu určím pomocí již nakreslených bodů. Samozřejmě se snažím dát všechny spoje na stranu pájení, chceme-li však mít dobré vodiče zemí a napájení, musíme obvykle nějaké spoje umístit i na stranu součástek. U složitějších desek se pak oběma stranám nevyhneme, i když používáme k rozvodu napětí pásky, které se do desky zapájejí jako součástky. Při návrhu je vždy dobré si očíslovat rohové body obvodu (1, 7, 8, 14), abychom se lépe orientovali. Spoje a pájecí body se kreslí jen symbolicky, bez ohledu na velikost. Širší spoje nebo plochy zemí a napájení kreslíme však tak, jak budou. Spoje, které jsme již zakreslili, si označíme ve schématu přeškrtnutím čáry u příslušného obvodu. Rovněž dopíšeme do schématu číslo špičky (vývodu) obvodu, kterou jsme použili. Při návrhu desky s plošnými spoji je nejdůležitější předem odhadnout, jak rozložit jednotlivé celky zařízení vzhledem k jejich propojení, k počtu spojů na konektory, atd. I když rozložení odhadneme, zakreslujeme do návrhu obvody po částech a podle toho, jak se návrh vyvíjí, přidáváme další. Snažíme se vždy využít všech hradel a klopných obvodů, abychom potom neměli dvě nepoužitá hradla někde, kde jsou okolo již samé spoje. Někdy nás návrh přinutí použít dvouvstupové hradlo jako invertor apod., protože by jinak zbylo a k invertoru je daleko – proto při úvahách o počtu obvodů na desce si musíme nechat nějaké hradlo do rezervy. Největší potíže při návrhu pak samozřejmě máme při posledních spojích, kdy už není nikde volná ulička. Nakonec doplníme návrh o blokovací kondenzátory napájení a návrh překontrolujeme. Při kontrole je dobré každý zkontrolovaný spoj znázornit ve schématu obtažením spoje červenou tužkou.

Jako pomůcku je vhodné mít při návrhu tři papíry. Na prvním si uděláme obrázky jednotlivých pouzder obvodů, aby bylo jasné, jak jsou obsazeny jejich špičky (vývody). Tyto obrázky je vhodné mít na stejném čtverečkovaném papíře, jaký používáme při návrhu a mít je dvoustranně a doplňovat je o další obvody. Na druhém papíře si označíme, jaké rozteče pájecích bodů jsou třeba pro tranzistor, pro odpory jednotlivých typů, konektory, atd. Vyšrafovanou plochou si pak označíme, jakou plochu na desce součástka zabere. Také tento papír neustále doplňujeme a zpřesňujeme. Třetí papír je na poznámky a můžeme ho po návrhu zahodit. Na něj si píšeme, která hradla jsou ještě volná, které špičky u dlouhého spoje (třeba nulování) ještě chybí propojit, atd. Problém plošných spojů pro amatéry je však nikoli v návrhu, ale ve výrobě. Proto pokládám za výhodnější používat univerzální desky s plošnými spoji a obvody „zadrátovat" podle schématu. S plošnými spoji budeme řešit jen taková zařízení, která chceme zhotovit několikrát.

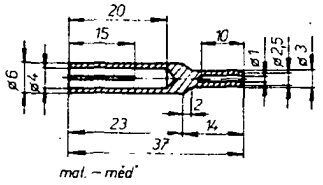
Výroba zařízení

K výrobě zařízení s integrovanými obvody mám pouze několik poznámek.

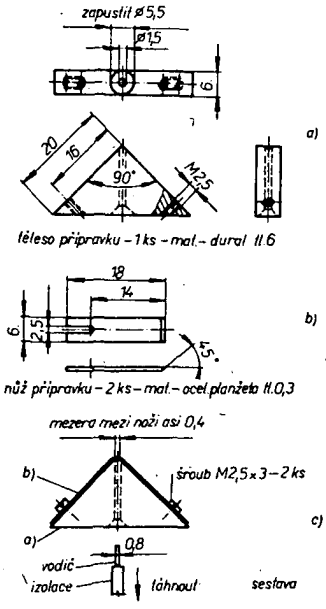
– Pájejte čistě a spolehlivě. Nepoužívejte pistolovou páječku. Sežente si nebo zhotovte mikropáječku s dutým hrotem na pájení integrovaných obvodů. Výkres hrotu je na obr. 36.

– Na odizolování drátů si vyrobte přípravek, který si přišroubujete buď ke stolu, nebo přímo na štípací kleště. Výkres přípravku je na obr. 37.

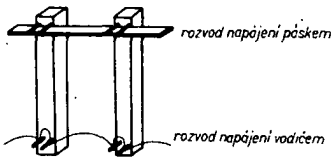
– Po zapájení součástek desku pečlivě prohlédněte lupou s co největším zvětšením.



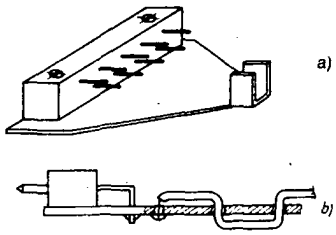
Obr. 36. Hrot páječky



Obr. 37. Přípravek k odizolování vodičů



Obr. 38. Rozvod napájecích napětí na konektory



Obr. 39. Připevnění kabelu; a) příchytka na kabel (z hliníkového plechu), b) uchycení kabelu provléknutím děrami v desce, nesoucí konektor

– Při pájení drátových spojů na univerzálních deskách a mezi konektory si označte různou barvou vodiče, které vedou napájecí napětí, „logické" vodiče, vodiče s jinými úrovněmi než TTL a oddělte je od sebe. Vázání vodičů do svazků zvětšuje vzájemné rušení, které se pak těžko hledá.

– Napájecí napětí v zařízení rozvádějte raději pásky laminátu s měděnou fólií šířky 10 mm než vodiči (obr. 38).

– Pečlivě zhotovte kabely. Velký počet závad v zařízeních je způsoben utrženým nebo ulomeným vodičem u konektoru kabelu (obr. 39).

– V zařízení se nesmí nic hýbat. Desky musí být dobře připevněny, vodiče musí být u konektorů v kabelech přichyceny. Součástky by měly být umístěny na desce tak, aby se při položení desky na stůl neohýbaly a nezkratovaly.

– Pozor na vedení zemí! Zesilovače ve zdrojích, operační zesilovače, komparátory, logické integrované obvody jsou vlastně vysokofrekvenční obvody s velkým zesílením a šířkou pásma řádu stovek MHz. Tyto skutečnosti musíte mít na paměti při spojování zemí. Při špatně vedených zemích stačí třeba jeden tranzistor, který spíná zátěž a zdroj rušení budete hledat hezky dlouho. Zvláště důležité jsou země ve zdrojích, u nichž kmitání bez dobrého osciloskopu nenajdete a stejně ho žádnými kondenzátory neodstraníte (obr. 40).

– Chcete-li mít zařízení spolehlivé, podrobte ho po uvedení do provozu testu za pomoci fénu a ledničky, vyměňte součástky, které vás zklamaly a pak nechte zařízení zapnuté dnem i nocí třeba týden – tím vyloučíte další vadné součástky.

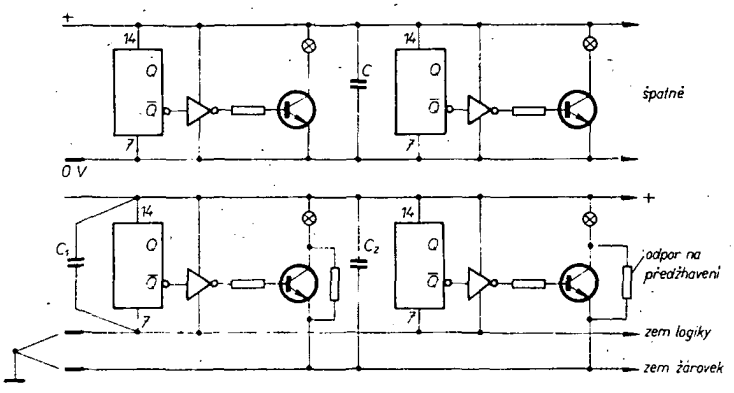
– Do hotového zařízení zasahujte co nejméně. Zbytečná manipulace s deskami a s konektory zmenšuje spolehlivost zařízení. Opravování zařízení způsobem kdy do vadného zařízení přendáváme desky ze stejného fungujícího zařízení, má často za výsledek dvě nespolehlivá zařízení.

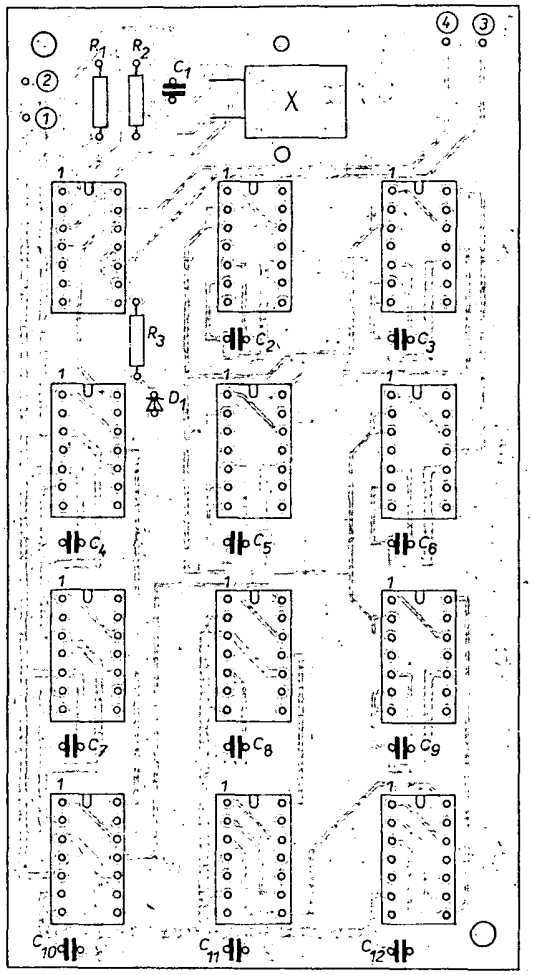
– Opět se učte ze zahraničních zařízení. Každá zdánlivá maličkost může být příště přínosem pro vaše zařízení.

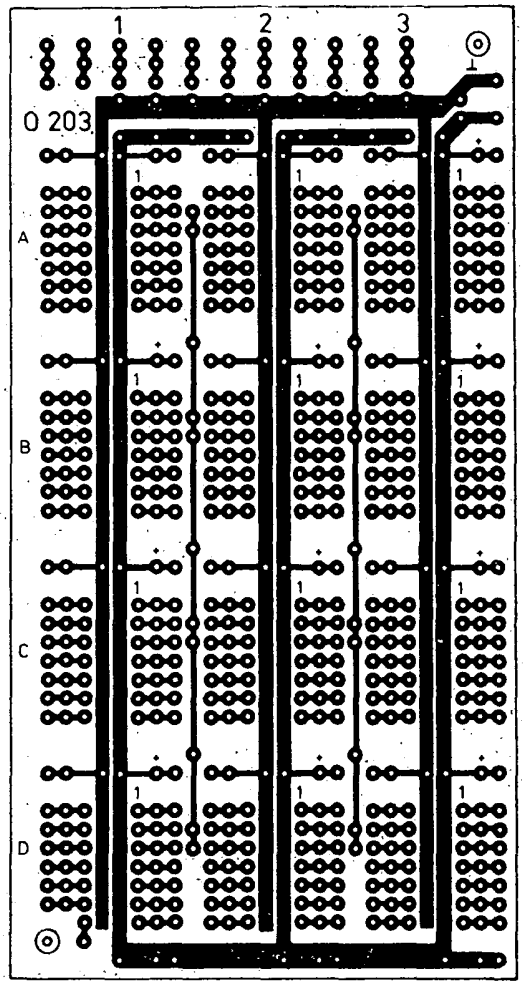
Oživování zařízení

Oživování zařízení je etapa, v níž musíme nalézt a odstranit všechny chyby, které jsme udělali v předcházejících etapách. Při hledání příčin špatné funkce zařízení není obvykle možné postupovat podle předem stanoveného návodu. Postup oživování je chaotickou kombinací několika metod a vyžaduje alespoň malý kolektiv, jednak proto, že některá měření je nutno realizovat ve dvou, a jednak je v kolektivu lepší možnost kontroly. Při tak duševně namáhavé práci, jako je oživování číslicového zařízení, dochází po určité době

Obr. 40. Oddělení „logické" země a země zátěží

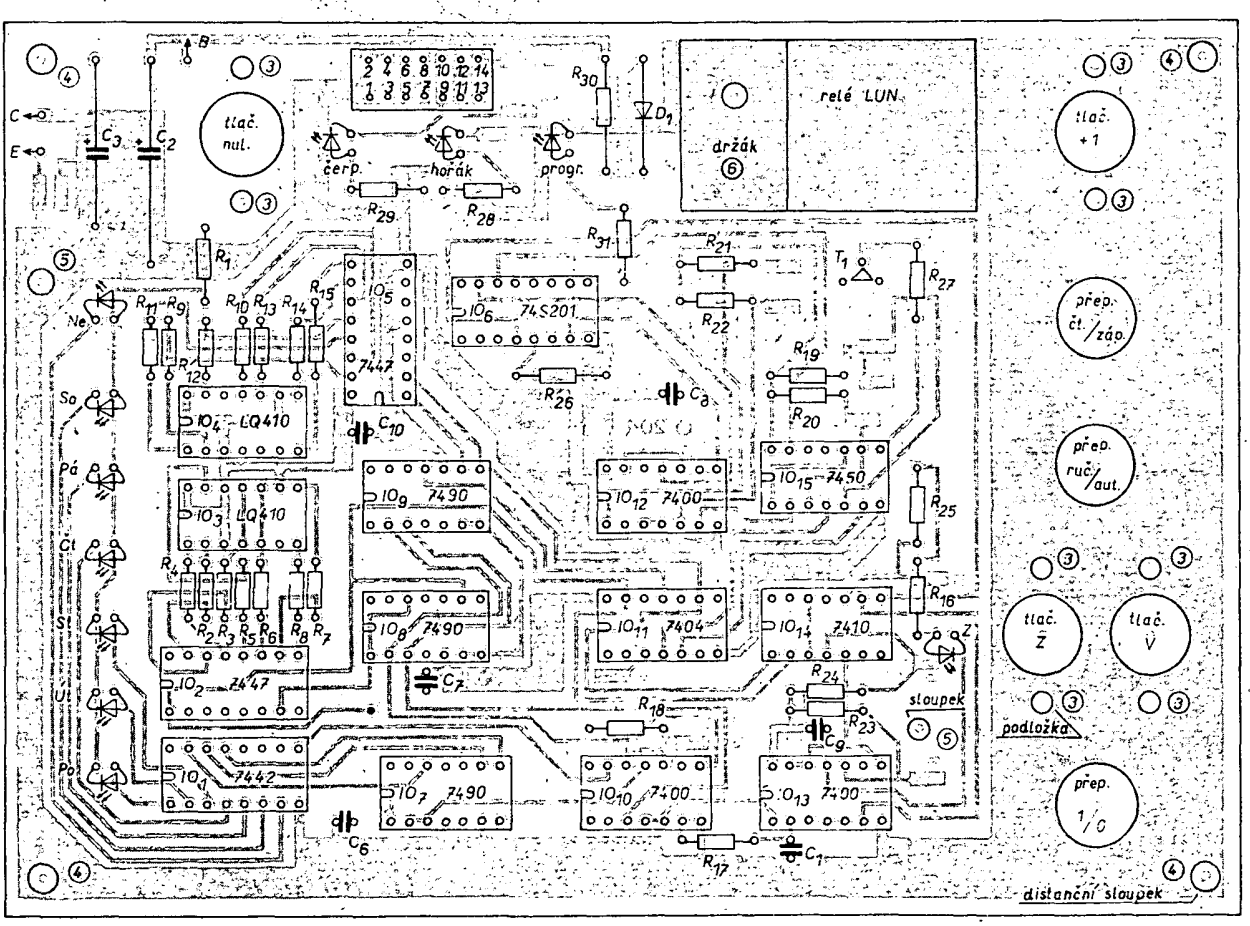


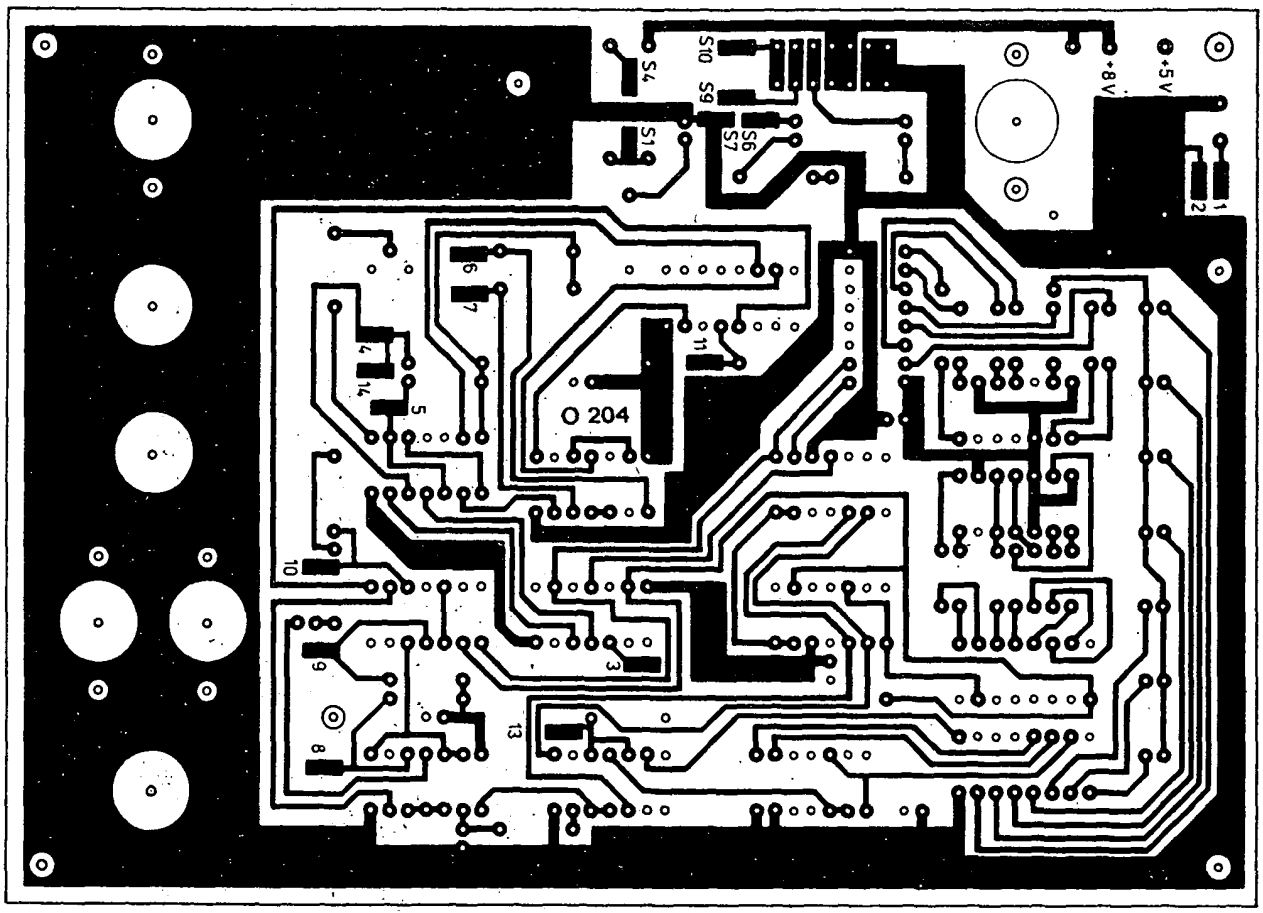


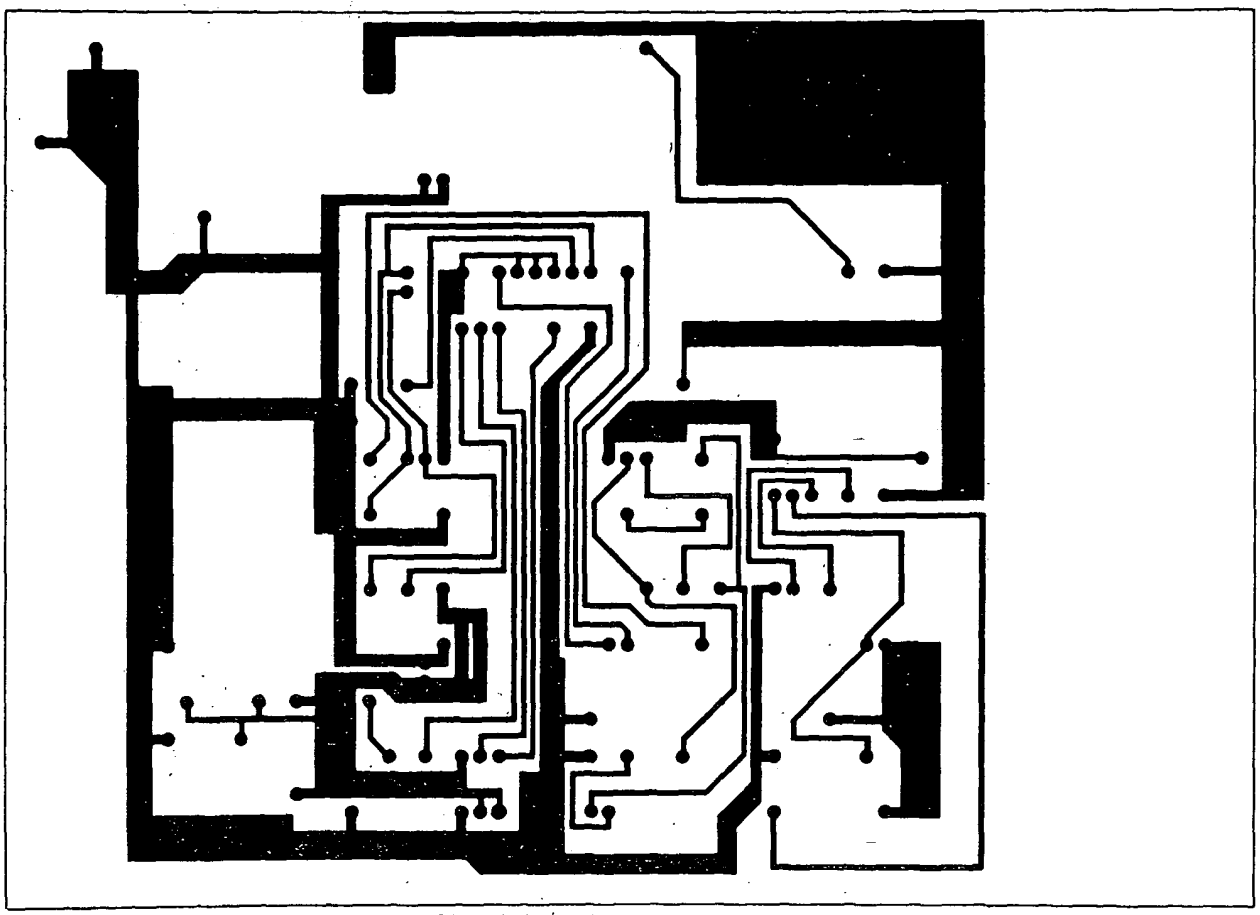


Obr. 48. Časovou základnu lze realizovat i na této univerzální desce s plošnými spoji (0203)

(Diody pro Po, Út a St je třeba zapojit obráceně, tj. prohodit katodu s anodou).







Obr. 49, 50. Deska s plošnými spoji programátoru (O204)

Obr. 52. Osazená deska programátoru je na 4. str. obálky

tím 8 až 10 V. Napětí se pak nastaví tak, aby na vstupu programátoru bylo při maximálním odběru proudu 7,8 až 8 V. Další stabilizátor je zařazen z těchto důvodů:

a) zajistí větší odolnost proti rušení, pronikajícímu ze sítě do zařízení;
b) umožní nastavit napětí na vstupu bez ohledu na úbytky na vodičích;
c) zajistí konstantní výkonovou ztrátu stabilizátoru +5 V v programátoru bez ohledu na kolísání sítě.

Kontaktem programátoru nemůžeme přímo ovládat motor hořáku, pracujícího se síťovým napětím 220 V. Proto musí být ovládací kontakt oddělen pomocí relé RP100. O způsobu instalace programátoru je nejlépe poradit se s odborníky na automatiky hořáků. Na obr. 53 je názorné schéma zapojení programátoru tak, jak bylo vyzkoušeno. Na obrázku je znázorněna možnost zálohování napájení baterií. Zálohování však nebylo prakticky odzkoušeno a nebylo řešeno dobíjení baterie. V současné době je nutné po výpadku napájení naprogramovat paměť znovu.

Zkušenosti

Po dvouletém provozu lze říci, že zařízení je spolehlivé, necitlivé na rušení a dobře obsluhovatelné. Úspora nafty za zimu byla asi 500 l a přitom není nutné starat se o vypínání a zapínání topení. Jako nevýhoda uvedeného řešení se jeví značný odběr energie (10 W) programátoru a ztráta programu při výpadku sítě. Tyto nevýhody by bylo možno odstranit použitím obvodů MOS a nebo CMOS. Menší odběr energie by umožnil zálohovat programátor baterií a zvětšila by se ekonomie zařízení. Při použití paměti CMOS by bylo výhodné rozšířit kapacitu paměti na 4 bity, aby bylo možno ovládat větší množství spotřebičů (rozvod topného media, světlo, vyhřívání akvária atd.).

Seznam dílů

Díl	Název	Kusů
1	deska programátoru	1
2	chladič	1
3	podložka tlačítka	8
4	distanční sloupek	4
6	držák relé LUN	1
7	kryt	1
8	panel	1
9 nebo 10	časová základna	1

Seznam součástek

Deska časové základny

R_1, R_2	odpor TR 151, 470 Ω
R_3	odpor TR 151, 270 Ω
C_1	ker. kondenzátor TK 744, 1 nF
C_2 až C_{11}	ker. kondenzátory, 10 nF
X	krystal 10 MHz
A_3	MH7404
B_3, C_3, D_3, C_1, D_1, D_2, C_2, B_1, A_1, B_2, A_2	MH7490
D_1	dioda LED, LQ100

Deska programátoru

LUN	relé lun 2621.40, 6 V	1 ks
přepínač síťový, s kovovou páčkou, 336–416		3 ks
tlačítko Isostat – krátké 10 × 4 mm		4 ks

Odpory

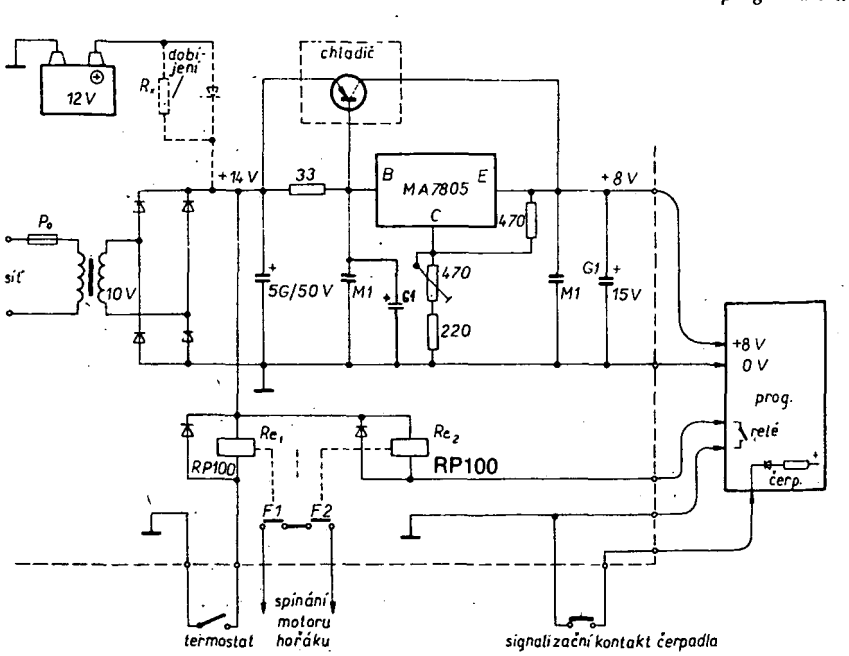
R_1, R_{16}, R_{28}, R_{29}	TR 151, 270 Ω
R_2 až R_{15}	TR 191, 560 Ω
R_{31}	TR 151, 120 Ω
R_{17}	TR 151, 100 Ω
R_{18} až R_{26}	TR 151, 1 kΩ
R_{27}	TR 151, 180 Ω
R_{30}	TR 151, 27 Ω

Kondenzátory

C_1	ker. kondenzátor TK 744, 15 pF
C_2	TE 984, 50 μF
C_3	TE 981, 20 μF
C_4, C_5	ker. kondenzátor TK 782, 100 nF
C_6 až C_{10}	ker. kondenzátor TK 744, 10 nF

Polovodičové prvky

D_1	dioda KY130/80
LED	dioda LQ100, 11 ks
IO_1	MH7442
IO_2, IO_5	MH7447
IO_3, IO_4	displej LQ410
IO_6	SN74S201
IO_7, IO_8, IO_9	MH7490
IO_{20}, IO_{12}, IO_{13}	MH7400
IO_{11}	MH7404
IO_{14}	MH7410
IO_{15}	MH7450
IO_{16}	MA7805



Obr. 53. Příklad instalace programátoru

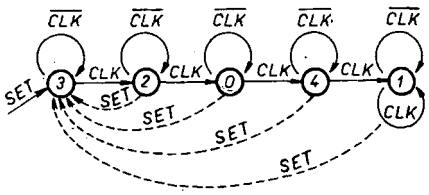
Navrhování sekvenčních obvodů

Sekvenční logické obvody navrhujeme obdobným způsobem jako kombinační logické obvody. V programátoru byly sekvenční obvody použity jako čítače, realizované v jednom pouzdře a proto jsme se jejich návrhem nezabývali. Nyní si stručně ukážeme postup výpočtu sekvenčního obvodu.

Problém: Navrhnout čítač, který čítá sekvenci 3 – 2 – 0 – 4 – 1 a po dosažení stavu 1 již není žádná změna stavu povolena. Nastavením je možno čítač opět nastavit do stavu 3.

Řešení

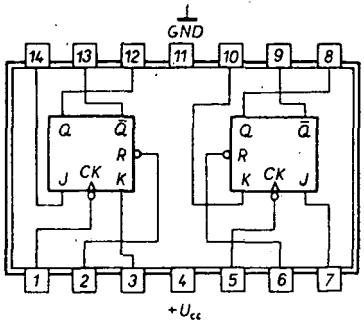
A. Znázorníme si funkci čítače stavovým diagramem. Stavový diagram je sestaven z koleček, do nichž píšeme stav sekvenčního obvodu. V našem případě to budou čísla, obvykle má však každý stav nějaký název. Vstupními signály je možno zajistit přechod sekvenčního obvodu z jednoho stavu do druhého. Přechody označíme čarou se šipkou a ke každé čáře napíšeme podmínku, kterou musí splnit vstupní signály pro přechod z jednoho stavu do druhého. Obvykle kreslíme



Obr. 54. Stavový diagram čítače 3-2-0-4-1

čáry pouze pro ty signály, které mění stav, ale my si nakreslíme čáry pro všechny vstupní podmínky. Podmínka CLK bude znamenat, že do našeho čítače přišel hodinový impuls, podmínka $\overline{CLK}$, že nepřišel. Podmínka SET bude znamenat asynchronní nastavení čítače do počátečního stavu. Stavový diagram je na obr. 54. Nastavení obvodu do výchozího stavu se kreslí pro přehlednost pouze šipkou, i když by mělo být znázorněno tak, jak je na obr. 54 čárkovaně.

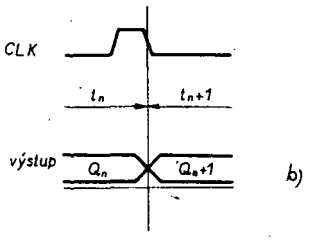
B. Dále určíme, jakými obvody budeme sekvenční obvod realizovat. Sekvenční obvod, který má pět stavů, musí být realizován nejméně ze tří klopných obvodů. Zvolíme klopné obvody typu J-K a nazveme je A, B a C. Použité obvody budou třeba 7473 s jedním vstupem J a jedním vstupem K (obr. 55).



Obr. 55. Zapojení obvodu 7473

Řádek	J	K	Q_{n+1}	
0	0	0	Q_n	stav se nemění
1	0	1	0	= 0
2	1	0	1	= 1
3	1	1	$\bar{Q}_n$	stav se změní 1→0 nebo 0→1

a)



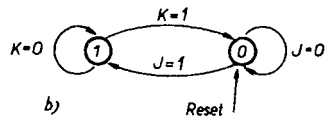
b)

Obr. 56. Pravdivostní tabulka obvodu 7473 (a) a definice označení Q_{n+1} (b)

C. Analyzujeme chování použitého obvodu. Pravdivostní tabulka klopného obvodu J-K je na obr. 56a. Na obr. 56b je definice označení Q_n a Q_{n+1}. Stav Q_n je stav před příchodem hodinového impulsu a Q_{n+1} stav po hodinovém impulsu. Z pravdivostní tabulky si můžeme odvodit podmínky pro vstupy J a K přechodu z jednoho stavu klopného obvodu do druhého. Tyto podmínky jsou na obr. 57a znázorněny tabulkou a na obr. 57b stavovým diagramem klopného obvodu 7473. Tabulku z obr. 57 budeme dále potřebovat!

Přechod	Postačující podmínka
1 → 0	K = 1
0 → 1	J = 1
0 → 0	J = 0
1 → 1	K = 0

a)

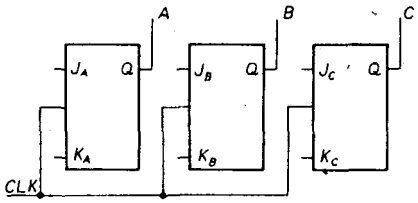


b)

Obr. 57. Postačující podmínky na vstupech J-K pro změnu stavu; tabulka (a) a stavový diagram (b)

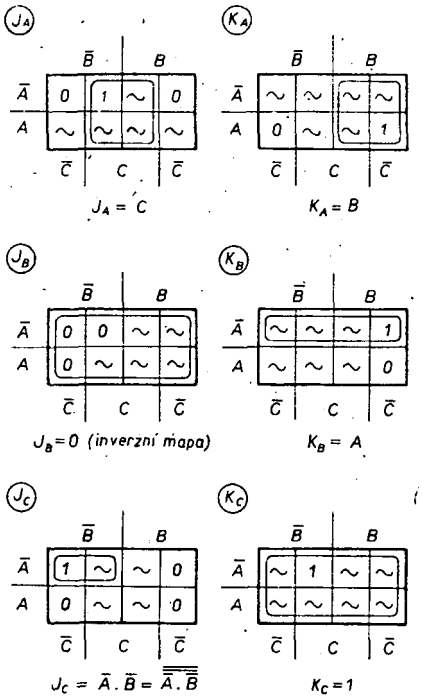
D. Napíšeme pravdivostní tabulku čítače skládajícího se z klopných obvodů A B C tak, jak je uspořádána na obr. 58. Pro přehlednost píšeme za poslední stav ještě následující stav. V našem případě se stav již nemění až do nastavení. Do prvního sloupce píšeme stav, do dalších tří stavy obvodů A, B a C. Dále musíme vyplnit nutné stavy na vstupech J a K pro všechny obvody. K tomu použijeme tabulku z obr. 57. Postup vyplnění je v tabulce znázorněn třemi položkami. Vlnovka opět znamená, že na stavu nezáleží. Tabulku si doplňte sami.

E. Realizace obvodů šesti logických funkcí, které dávají signály J a K pro jednotlivé klopné obvody, svěříme již osvědčeným Karnaughovým mapám. Mapy jsou na obr. 59 a realizace čítače na obr. 60. Je samozřejmé, že pro každý stav větší než 4 můžeme políčko mapy doplnit vlnovkou.

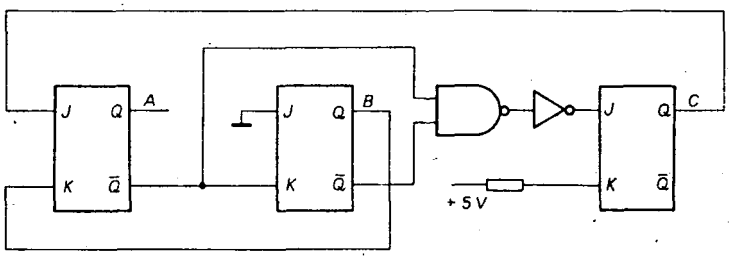


Dek.stav	Bin. stav			Vstupy A		Vstupy B		Vstupy C	
	C	B	A	J_A	K_A	J_B	K_B	J_C	K_C
3	0	1	1	~	1	~	0	0	~
2	0	1	0	0	~	~	1	0	~
0	0	0	0	0	~				
4	1	0	0	1	~				
1	0	0	1	~	0			0	~
1	0	0	1						

Obr. 58. Pravdivostní tabulka čítače 3-2-0-4-1, vyplněná pomocí tabulky z obr. 57



Obr. 59. Mapy sestavené podle tabulky na obr. 58



Obr. 60. Obvodová realizace čítače 3-2-4-0-1

Literatura

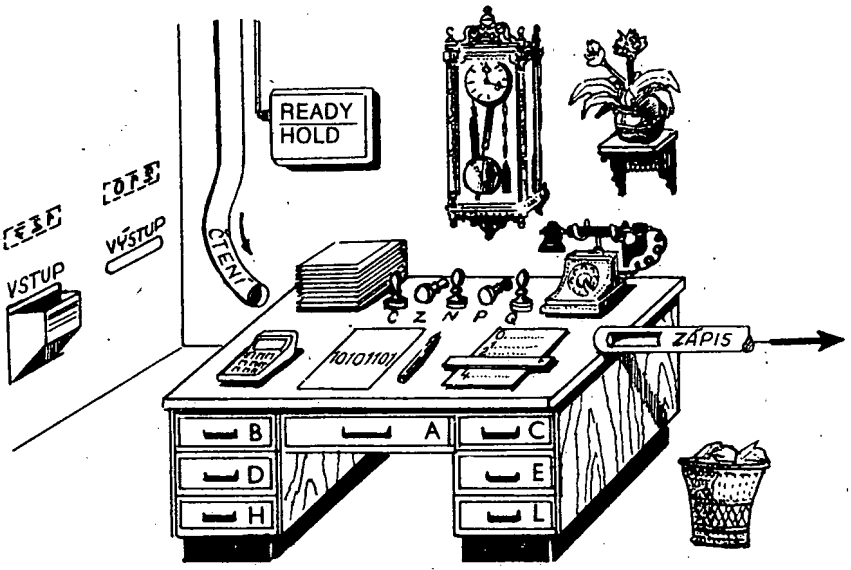
Computer design 10, č. 2, s. 49 až 53.

ŘEŠENÍ PROGRAMÁTORU S MIKROPROCESOREM

Úvod

Mikroprocesor je ideální hračka. Je malý, spolehlivý a levný. Sám o sobě je sice hloupější než šlapací autíčko, ale můžeme ho naučit skoro všechno. Nejprve mu musíme rozumět. Nemějte strach, že se vám to nepodaří. Ve stejné situaci jako vy se v současné době nalézají statisíce vývojových, technických a řídících pracovníků, kutilů i dětí na celém světě. Přitom je třeba připomenout, že vy, kteří považujete kalkulačku za běžnou věc, to máte jednodušší než technik, který umí přemýšlet tak, jak se to několikaletou praxí naučil. Mikroprocesory nejsou revolučními prvky elektroniky proto, že by nějak podstatně měnily principy, na kterých byla dosud založena technika počítačů, avšak proto, že umožňují prakticky každému tuto techniku používat. Kdo nemá čas se podrobně zabývat mikroprocesorem a pomocnými obvody, které potřebuje k určité aplikaci, může si koupit celý mikropočítač a psát si třeba jen programy. Kdo má zájem, může si vyvinout svůj systém, „ušitý" na jeho aplikaci. Předem chci upozornit ty, kteří budou mít zájem a možnost pracovat s mikroprocesory, na důležitou věc. Nenechte svoje mikropočítače jenom počítat, bylo by jim to určitě líto, nechte je něco dělat. Ať se třeba snaží řídit vláčky, hlídat rybičky, hrát hudbu nebo kreslit a mluvit. Počítače nám přece musí pomáhat, kamarádit s námi a ne nás zasypávat čísly, která ani nestačíme číst, natož používat. Proto, abychom v budoucnu mohli svěřit mikroprocesoru podobné úkoly, musíme vědět, jak pracuje. Já už jsem to zjišťoval a proto se s vámi rád rozdělím o výsledky svého pátrání.

Jednou, když jsem přemýšlel o tom, jak to ten brouček dělá, představil jsem si dveře do kanceláře, na kterých bylo napsáno I8080. V kanceláři pracoval úředník, který se zdál na prvý pohled hloupý, ale bylo vidět, že svou práci dělá naprosto přesně. Když jsem se podíval po kanceláři, bylo mi hned jasné, že je to tak trochu vynálezce-amatér. Patrně brzy zjistil, že by pouze s vlastní hlavou svou práci nezvládl a proto si kancelář vylepšoval a vylepšoval. Seděl u psacího stolu, který měl sedm zásuvek, kterým říkal registry, a aby si je nepletl, popsal si je písmeny A, B, C, D, E, H a L (obr. 61). Prostřední zásuvku, A, používal na střádání posledního výsledku své práce a říkal jí AKUMULÁTOR. Zásuvky po stranách měly jednu velkou výhodu. Bylo je možno vysunout po jedné a nebo také po dvojicích, použil-li k tomu obě ruce. Vytáhl-li dvojici registrů, třeba H a L, mohl z nich vzít dva papíry najednou, nebo tam dva vložit a to už přece stálo za to. Na stole vlevo měl kalkulačku, které říkal ARITMETICKO-LOGICKÁ JEDNOTKA (ALU – byl to Angličan). Na stole vpravo měl vždy papír, na kterém byl napsán postup práce, kterému říkal PROGRAM. Aby nezapomněl, co má dělat teď a co potom, měl na programu položeno pravítko, říkal mu ČÍTAČ PROGRAMU (PC) a vždycky, když si přečetl z programu INSTRUKCI, co má dělat, jednoduše posunul pravítko na další řádek. Na



Obr. 61.

stole měl ještě telefon, kterým mu volal jeho vedoucí, když měl pro něho nějakou důležitější práci. Protože každé zavolání znamenalo přerušení jeho současné práce, říkal zvonění telefonu SIGNÁL O PŘERUŠENÍ (INTERRUPT SIGNAL). Jenže zkuste přerušit rozdělanou práci a začít jinou, při které může zazvonit vedoucí, že má práci ještě důležitější. Aby v tom měl pořádek, zavedl si na stole hromadu papírů a říkal jí SKLÍPKOVÁ PAMĚŤ (STACK). Tato paměť má totiž tu výhodu, že si nemusíte pamatovat, kam jste co uložili. Máte-li důležitější práci, uložíte vše, co máte na stole, na hromadu a začnete dělat novou práci. Musíte-li i tuto práci přerušit, přidáte nevyřízené papíry opět na hromadu. Až tuto práci ukončíte, vezmete si z vrchu hromady přerušenou práci, až ji doděláte, vezmete si z hromady další práci a když ji uděláte, můžete pokračovat v práci podle programu od místa, kde jste byl přerušen. Úředník měl ještě jeden problém. Některé instrukce, které měl vykonávat, byly závislé na tom, jak dopadlo vyplnění předešlé instrukce. Aby si pamatoval, jak dopadlo plnění předešlých instrukcí, sehnal si pět razítek, na nichž bylo napsáno C, Z, N, P, Q, dal si je před sebe a když si potřeboval zapamatovat výsledek nějaké důležité operace, položil nebo postavil razítka podle výsledku operace. Těmto razítkům říkal STAVOVÝ REGISTR (SR). Jak práce přibývalo, nebylo už kam dávat žádosti, stížnosti, pochvaly a také mezivýsledky složitého vyřizování. Proto si úředník zařídil místnost o patro výše, kde bylo 64 000, možná o trochu více, přihrádek a každá měla své číslo – ADRESU. Ve své kanceláři měl potrubní poštu a potřeboval-li něco založit, udal jenom číslo přihrádky a do otvoru potrubí, kde bylo napsáno ZÁPIS (WRITE), vložil příslušný spis. Spis se sám uložil do příslušné přihrádky v horní místnosti. Této místnosti říkal PAMĚŤ (MEMORY). Naopak, potřeboval-li vědět, co je v přihrádce třeba č. 12 275, udal potrubní poště toto číslo jako adresu a spis mu vypadl na stůl otvorem ČTENÍ (READ).

Co by to bylo za úředníka, kdyby nepracoval pro lidi. Aby se mu nenahrnuli do kanceláře, rozdal vždy lístečky (měl jich 256) a pak stačilo, aby se v čekárně rozsvítilo číslo (ADRESA VSTUPU) a příslušný občan vložil svou žádost do schránky VSTUP (IN). Měl-li úředník žádost vyřízenu nebo zamítnutu, rozsvítil v čekárně zase číslo (ADRESA VÝSTUPU) a řádně zpracované papíry vložil do štěrbiny VÝSTUP (OUT) a příslušný občan si je převzal.

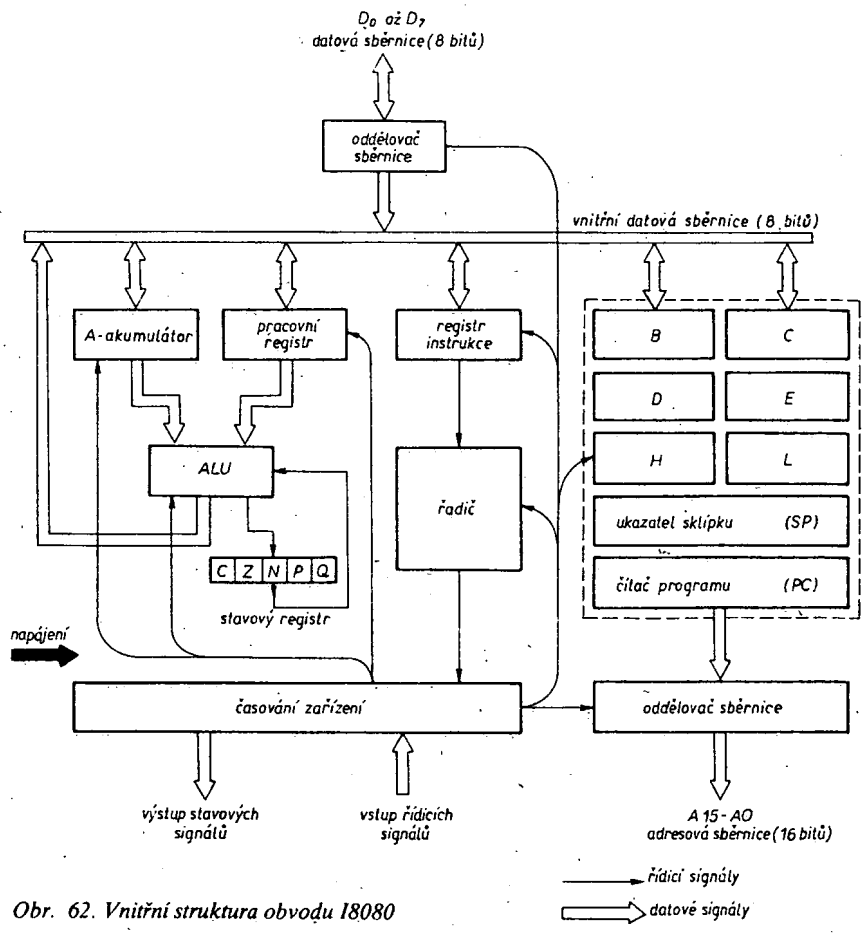
Zařízené to měl pěkně a tak jsem byl zvědav, jak bude pracovat. Ráno přišel, posunul pravítko PC na řádku označenou nula, této operaci říkal NULOVÁNÍ (RESET). Přečetl si první instrukci a pravítko (PC) posunul na řádku označenou 1. Potom instrukci vykonal. Vykonání instrukce měl přesně spočítáno. Naproti na stěně měl HODINY (CLOCK) a těch se držel. Bylo zajímavé sledovat, jak pracuje přesně podle kyvadla. Na každé kývnutí kyvadla provedl jeden OPERAČNÍ KROK. Tak třeba vložil adresu pro paměť, založil papír do registru D atd. Několik operačních kroků dohromady (3 až 4) tvořilo OPERAČNÍ CYKL a těch bylo potřeba na vykonání jedné instrukce 1 až 5. Takže INSTRUKČNÍ CYKL neboli doba, za kterou vykonal jednu instrukci, byla různě dlouhá, podle složitosti instrukce. Někdy se stalo, že se potrubní pošta zpozdila a pak byl úředník rád, neboť čekal (stav WAIT), až paměť rozsvítí žárovku PŘIPRAVEN (READY) a pak si teprve převzal data. Stejným způsobem čekal na VSTUP a VÝSTUP. Další zajímavost plynula z toho, že paměť v horní místnosti nebyla pouze pro něj. Proto občas dostal příkaz, aby na chvíli počkal (HOLD) a dovolil někomu jinému použít potrubní poštu k paměti. Potom mohl pracovat zase dále.

Instrukce v programu, který měl na stole, byly různé. Byly to instrukce pro předání obsahu z jedné zásuvky do druhé (MOV B C – předej to, co je v C, do B). Nebo obouruční instrukce, kdy se měnily současně obsahy zásuvek D, E a HL (XCHG). Dále to byly aritmetické instrukce, kdy třeba vytáhl papír ze zásuvky B, opsal číslo, které na něm bylo napsáno do kalkulačky, přičetl jedničku a na papír napsal výsledek a vložil ho zpět do zásuvky B a starý obsah zásuvky hodil do koše (INR B). Dále prováděl s čísly různé logické operace a podle výsledků si nastavoval razítka ve stavovém registru.

Někdy došel k takové instrukci, která mu říkala, aby přesunul pravítko někam úplně jinam a začal provádět například řádek 127. Jindy byl tento SKOK v programu PODMÍNĚN stavem některého z razítek. Nebylo-li razítko nastaveno tak, jak mělo být, přešel na další řádek, bylo-li nastaveno, říkala instrukce, kam má pravítko posunout.

Zajímavé bylo také, jak adresoval paměť. Do jedné zásuvky se mu vešlo menší číslo, než bylo potřeba na adresu až 64 000. Proto někdy používal adresu, kterou opsal ze zásuvek H a L dohromady (nebo z jiné dvojice zásuvek).

Co všechno uměl, to si můžete přečíst v seznamu instrukcí mikroprocesoru 18080. Teď se však obávám, že když vám řeknu, že udělal milióny kroků za sekundu v kanceláři velké několik čtverečních milimetrů, nebu-



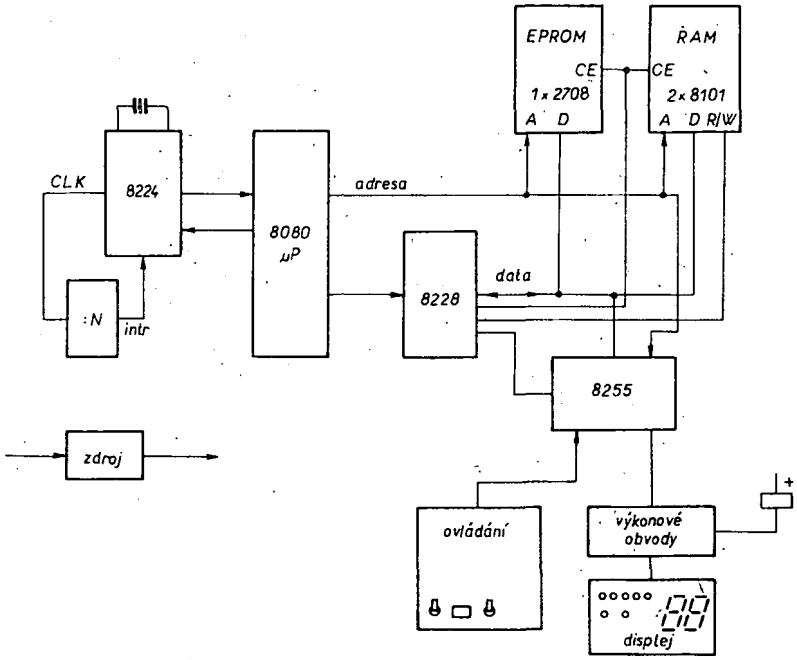
Obr. 62. Vnitřní struktura obvodu I8080

dete mi to všechno věřit a budete si to chtít zkusit. A o to mně právě jde. Připomeňte si ještě pro úplnost, že skutečný obvod I8080 má program, sklípkovou paměť, paměť dat i vstupy a výstupy adresované pomocí společné adresové sběrnice o šestnácti vodičích. Abychom věděli, kde je v této paměti sklípková paměť, má procesor vnitřní registr, UKAZATEL SKLÍPKU (SP), který je šestnáctibitový. Když do sklípku uložíme data, posune se ukazatel směrem dolů, když data odebíráme, posune se směrem nahoru. Na obr. 62 je pro srovnání s výkladem nakreslena vnitřní struktura mikroprocesoru I8080.

Postup návrhu zařízení s mikroprocesorem

Předpokládejme, že máme k dispozici mikroprocesor I8080 se základními obvody hodin, řízení sběrnice a zesílenými signály adresové a datové sběrnice. Dále předpokládejme, že máme vyřešeno připojování pamětí RAM a ROM (EPROM) ke sběrnicím a vstupních a výstupních signálů, třeba přes obvody 8212. Říkám to záměrně stručně, protože realizace tohoto základního systému je značně závislá na typech obvodů, které budou k dispozici, a také proto, že schémat zapojení až do tohoto bodu problému je v literatuře mnoho. Je možno říci, že návrh zařízení, máme-li předem dán typ mikroprocesoru, začíná až po vyřešení tohoto základního systému. Při konečné revizi celého systému pak můžete i tento základní systém zjednodušit, ale pro etapu ladění programů a oživování systému potřebujeme mít alespoň fungující základ, který je odolný proti přetížení na sběrnicích. Také kapacita pamětí ROM a RAM bývá v této etapě větší, než je zapotřebí pro konečné řešení. Součástí základního systému bývají také obvody a programové vybavení pro krokování a sledování průběhů programů. Předpokládám-li, že takový systém mám k dispozici, pak počítám ještě s tím, že kdo chce navrhovat zařízení s mikroprocesorem, perfektně rozumí funkci tohoto systému jak po obvodové, tak po programové stránce. U návrhů zařízení s logickými obvody je totiž vždy možnost převzít nějakou část, i když neznáme přesně do detailu její funkci. U systémů s mikroprocesory je funkce dána synchronizovanou spoluprací programů a obvodů systému a přebírání schémat a programů z podobných systémů je tak náročné na detailní rozbor, že se většinou vyplatí vyřešit vlastní systém i s programy. Samozřejmě zde nemám na mysli systémy pro edici, překlad a ladění programů, ale systémy, u nichž mikroprocesor něco řídí, což by mělo být jeho hlavním posláním.

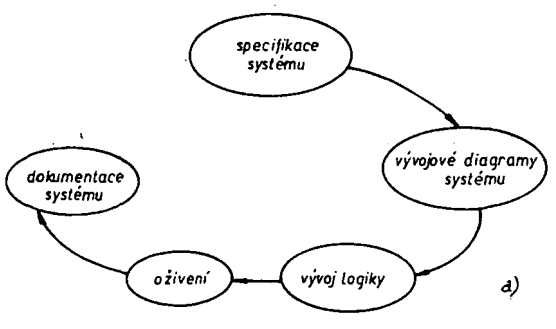
Vlastní postup při návrhu zařízení s mikroprocesorem se příliš neliší od postupu, který jsem popsal na příkladu návrhu programátoru. Na obr. 63a je znázorněn postup návrhu logického zařízení a na obr. 63b postup návrhu zařízení s mikroprocesorem. První etapa, kde se oba postupy liší, má za úkol rozhodnout, jaké funkce budou v zařízení plnit programy, a jaké logické obvody. Abychom využili procesor, který bude základem systému, budeme se samozřejmě snažit svěřit co nejvíce práce programům. Pro první přiblížení k řešení systému je to správný krok. Budeme-li o plnění všech funkcí zařízení uvažovat tak, jako bychom je řešili programy, ujasníme si velmi dobře funkci zařízení. I když potom z důvodů složitého programu, malé rychlosti takového řešení, velkých požadavků na paměť nebo složitosti oživování a provádění změn v systému převedeme část funkcí zpět do logiky, bude na základě tohoto postupu zajištěno plné využití procesoru a dobrá návaznost programů na logiku. Opačný postup, navrhnout logiku řízenou procesorem a pak ji zjednodušovat, nedoporučuji, protože naše myšlení je zatíženo dosavadními způsoby návrhu zařízení a vede k násilnému přizpůsobení logiky a programů a k nevyužití všech možností mikroprocesorové techniky.

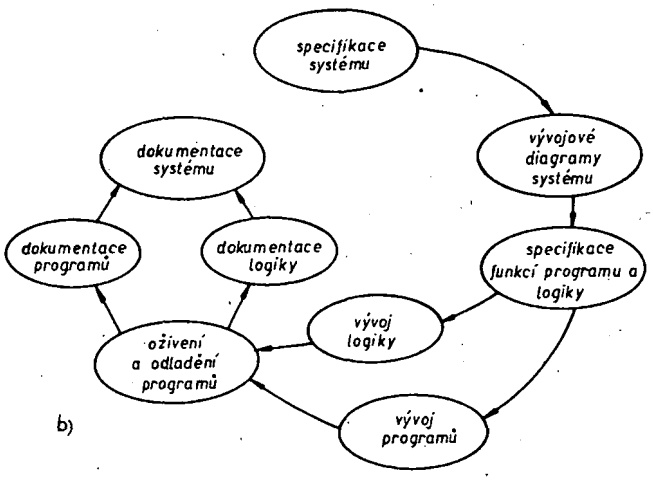


Obr. 64. Blokové schéma programátoru

Další kroky řešení jsou pak značně závislé na technických prostředcích, které máme k dispozici. Při navrhování zařízení s logickými obvody bylo potřeba pro návrh, vývoj a oživení mnoho pomůcek a přístrojů. Pro stejnou práci s mikroprocesory potřebujeme mnohem víc takových pomůcek a většinu z nich již není možno realizovat amatérsky tak, jako dřív. Potřebujeme programovat a mazat paměti EPROM, simulovat paměti ROM pamětmi RAM, psát a opravovat programy, ladit programy atd. Mikroprocesor je jako ledovec. Těch 10 %, co je vidět, je v současné době středem zájmu techniků a řídících pracovníků. O tom, že existuje ještě 90 % celého problému, se přesvědčíme při skutečné realizaci prvních systémů. Rozvoj mikroprocesorů musí být proto provázen několikanásobně prudším rozvojem v oblasti konstrukčních součástek, kabelů a příslušenství k nim, přídavných zařízení a prostředků pro vývoj mikroprocesorových systémů. Jinak zůstane mikroprocesor, tak jako dosud, pouze hlavním bodem seminářů, konferencí a článků. Na druhé straně je třeba říci, že část toho, co potřebujeme ke stavbě amatérských mikropočítačů, si můžeme přizpůsobit ze stávající součástkové základny, i když výsledek nebude vždy spolehlivý a estetický. Větší problém bude s přídavnými zařízeními, ale věřím, že právě amatéři přijdou na řešení, která umožní budoucí rozvoj nového oboru zájmové činnosti – mikropočítačů.

Vraťme se nyní po tomto úvodu k otázce řešení programátoru na bázi mikroprocesoru I8080. Kdybychom chtěli realizovat programátor, který byl plnil stejnou funkci jako uvedený typ, potom bychom potřebovali obvody podle blokového schématu na obr.





Obr. 63. Postup návrhu zařízení

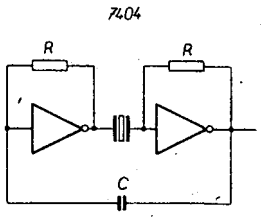
64. Obvody 8224, 8228 a 8080 tvoří základní systém procesoru. Dělička slouží pro generaci signálu přerušení. Přerušení je určeno pro program, který udržuje v zařízení časovou informaci o dnu a hodině. Paměť EPROM slouží pro uložení programu a paměť RAM pro uložení dat. Vstup a výstup zajišťuje obvod 8255. Výkonové obvody pak řídí displej a relé. Ovládací prvky jsou testovány přes vstupní obvody 8255. I když mikroprocesory jsou ve světě levné, přece jen je každému jasné, že na uvedenou aplikaci by jich bylo škoda. Před stejným problémem stojí výrobci měřicích přístrojů, automobilů a spotřebního zboží. Nejde přece jen o to mikroprocesor použít, ale účelem jeho použití musí být zvětšení komfortu obsluhy, úspora energie, větší výkonnost nebo schopnost plnit nové funkce. Proto není možné u jednoduchých zařízení prostě předělat vnitřek na řízení mikroprocesorem. U našeho programátoru by bylo nutné nalézt ještě další aplikace programového řízení v domácnosti, specifikovat nový systém a pak ho realizovat. Bylo by možné řídit i další spotřebiče. Dále by bylo možné, aby zařízení hlídalo stav odběru elektrické energie a informovalo nás o tom, jak s energií doma hospodaříme. Na systém, dá se říci domácího počítače, by bylo možné připojit bezpečnostní zařízení proti vloupání nebo požáru. Systém by bylo možné využít i jako paměť pro rozvrh školy, pro úkoly, které máme splnit, a podobně. Teprve potom, až by systém odpovídal svou cenou svému výkonu a prospěchu, teprve potom by aplikace mikroprocesoru byla oprávněná.

KUCHAŘKA TTL

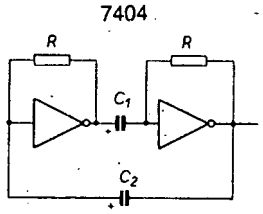
Ing. Eduard Smutný

Kuchařka obsahuje obvody, které často používám. Obvody nepotřebují žádné vysvětlení. Používání různých zapojení je značně závislé na vlastních zkušenostech a na oblibě určitého zapojení. Proto není dost dobře možné pokládat některá zapojení za horší a jiná za lepší. Rozhodujícím kritériem by měla být jednoduchost, spolehlivost funkce a součástková dostupnost. Každý, kdo pracuje s logickými obvody, by měl podobnou kuchařku mít. Sestavit ji lze např. takto: nejprve sbírám do sešitu všechna zajímavá zapojení z časopisů a z dokumentací zařízení výpočetní techniky. Poznamenám si důležité údaje z původního článku a údaje pro vyhledání původního pramenu informace. Když potřebuji podobný obvod použít, vyzkouším ho, případně něco měním a „chodí-li" obvod dobře, zařadím ho do kuchařky.

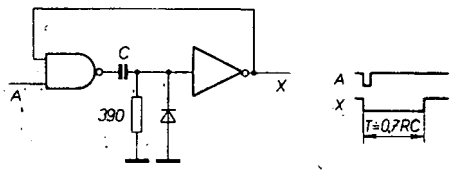
V kuchařce není dobré mít velké množství zapojení řešících jeden problém, je nutné vybrat jedno nejvhodnější zapojení a častým používáním na něm „vychytat" možné chyby. Snaha používat pro standardní obvody stále nová a nová zapojení může zanést do návrhu zařízení další chyby, které se špatně hledají, neboť funkci standardních obvodů bychom měli vždy věřit. Obvykle jsou všechna tato zapojení obecně známá, problém je v tom mít je pohromadě po ruce – a proto je předkládám těm, kterým je toto číslo AR určeno – mladým a začínajícím kybernetikům a ostatním zájemcům o číslicovou techniku.



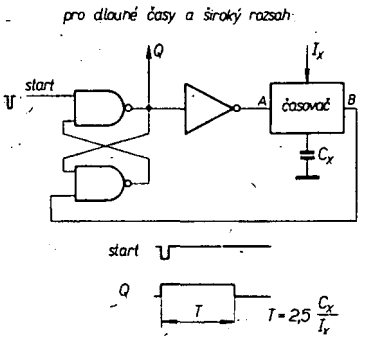
Obr. 1. Generátor hodinového signálu s krystalem (R = 470 Ω až 1 kΩ, C = 470 pF až 10 nF, krystal 1 až 20 MHz)



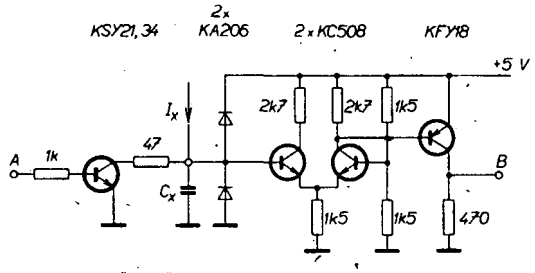
Obr. 2. Generátor RC (R = 470 Ω, C = 100 pF až 100 μF, $C_1 = C_2$)

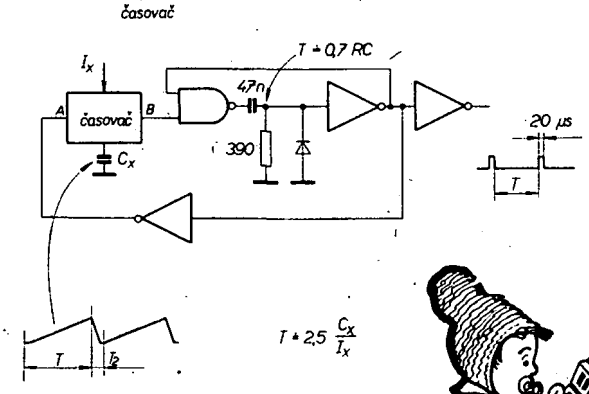


Obr. 4. Monostabilní obvod

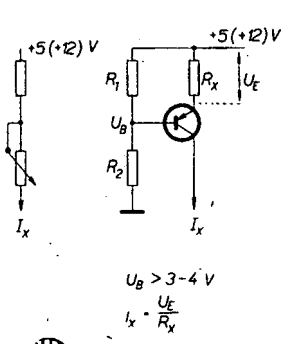


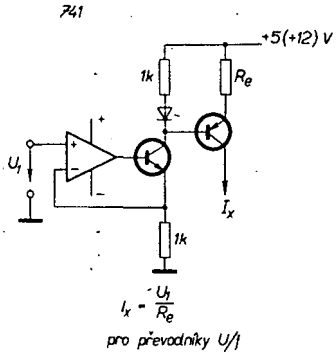
Obr. 5. Monostabilní obvod řízený proudem (možno řídit z převodníku D/A)



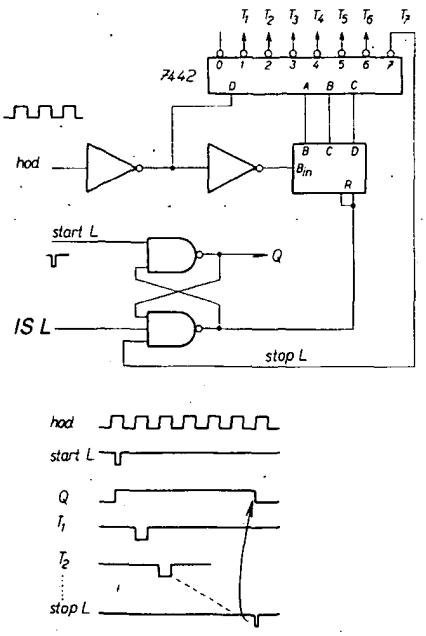


Obr. 3. Generátor řízený proudem (pro nižší kmitočty, laditelný, obdoba NE555)

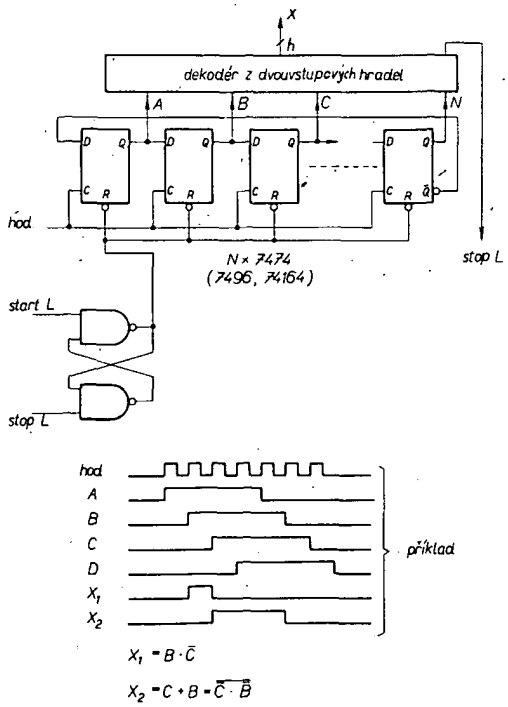




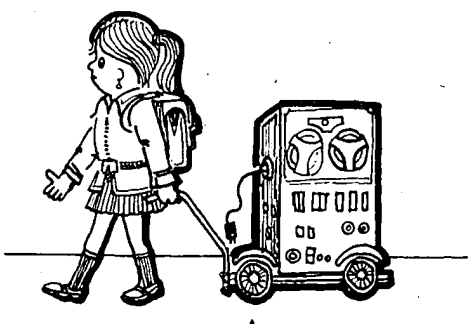
Obr. 6. Zdroje proudu pro časovač

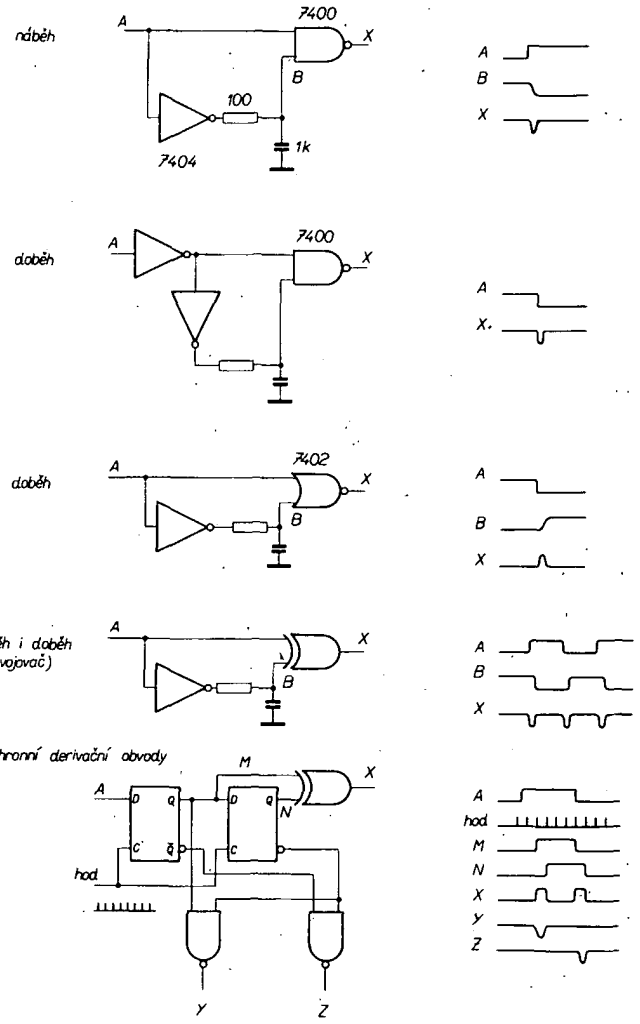


Obr. 7. Mnohofázová časová základna (lze zkrátit připojením „stop L" na nižší výstup, lze prodloužit obvodem 74154 na 14 výstupů, IS L – nulování po zapnutí)

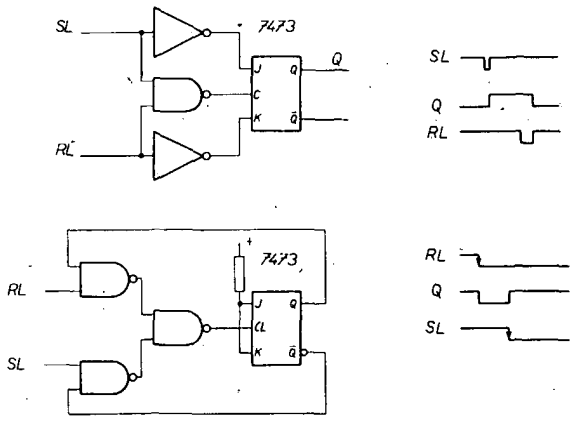


Obr. 8. Časová základna pro libovolné průběhy (Johnsonův čítač); výhody: nemá hazardní stavy, má jednoduchý dekodér, délka sekvence je libovolná, s obvody 74S74 pracuje po 50 ns

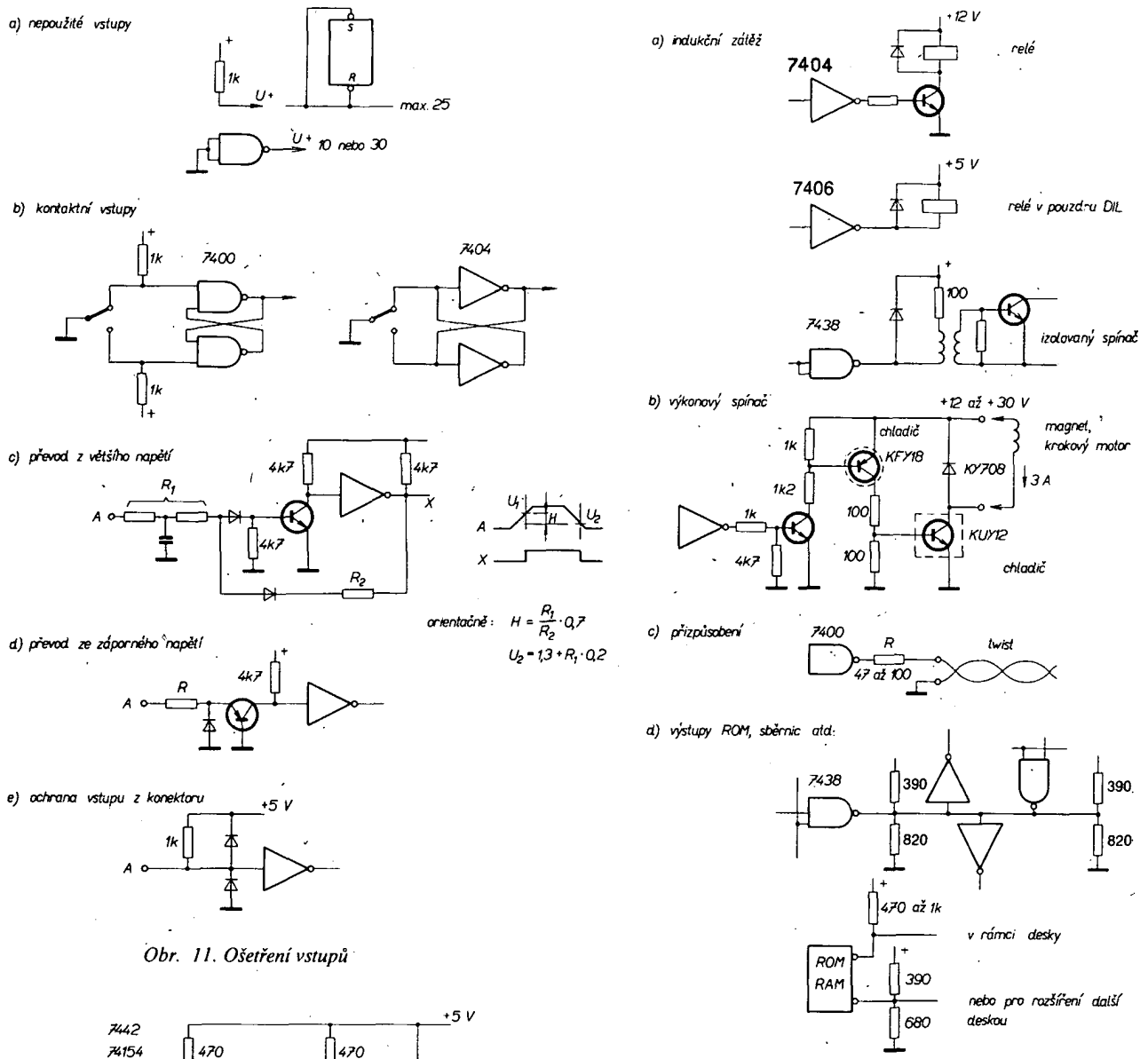




Obr. 9. Derivační obvody

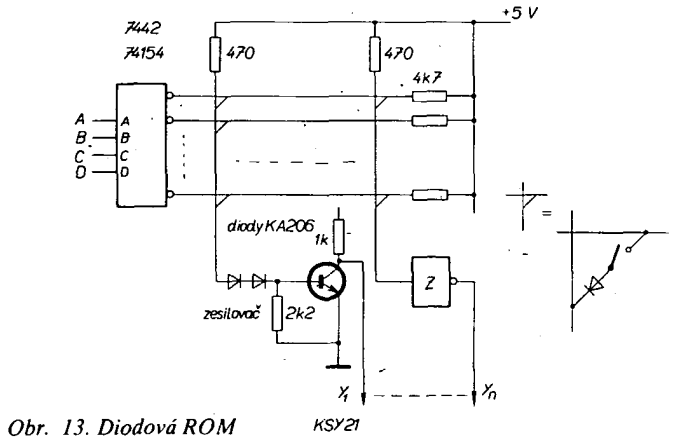


Obr. 10. Klopné obvody ovládané hranou

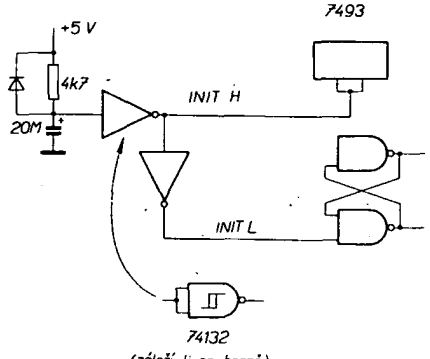


Obr. 11. Ošetření vstupů

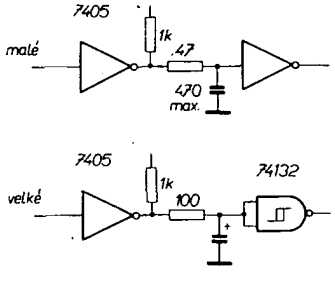
Obr. 12. Ošetření výstupů



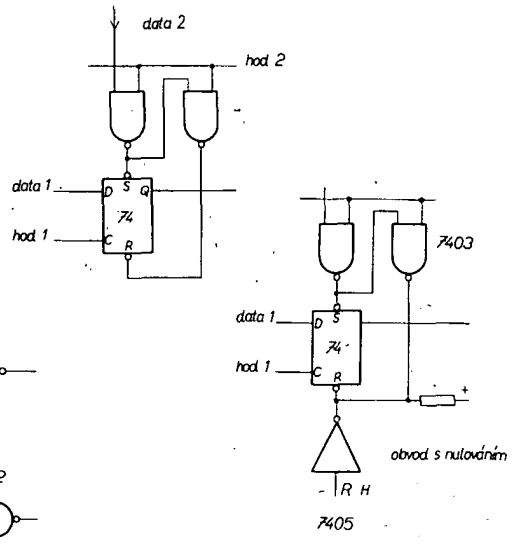
Obr. 13. Diodová ROM



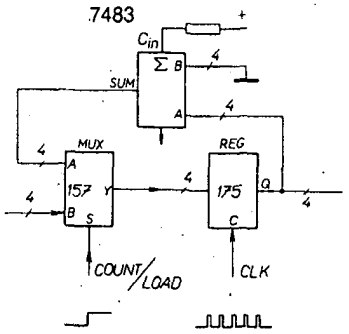
Obr. 14. Nulování po zapnutí



Obr. 15. Zpoždění signálu



Obr. 16. Rozšíření paralelního vstupu



Obr. 17. Náhrada čítače se synchronním čítáním a synchronním paralelním nahráním (74161)

Aktuality z oddělení elektroniky a kybernetiky Městské stanice mladých techniků Domu pionýrů a mládeže hl. města Prahy

V Městské stanici mladých techniků je po čtvrtém stěhování zapojován starší a těžší bratr mikropočítačů – počítač ZUSE 23. I když se tento počítač svým příkonem (6,5 kW) a rozměry (viz obr. 2 a 3 na 2. str. obálky časopisu) nemůže rovnat mikropočítačům, bude ukázkou historie samočinných počítačů za období čtrnácti let.

V současné době probíhá v oddělení elektroniky a kybernetiky diskuse o konstrukci osobního počítače. Vzhledem k záměrům n. p. TESLA a vzhledem k ceně mikroprocesoru 8080 (12,30 DM) byla dohoda o CPU snadná, větší problémy jsou s výběrem sběrnice a vhodného konstrukčního řešení. Mechanická konstrukce nemůže být amatérského původu, neboť vedení desek od konektorů a spolehlivé a levné konektory jsou důležitou podmínkou spolehlivosti celého zařízení. Z těchto důvodů byla vybrána jako nejvhodnější mechanická konstrukce stolní kalkulačky ELKA (BLR).

Touto volbou mechanické konstrukce však bylo ovlivněno řešení sběrnice, neboť ELKA má na konektorech pouze 41 špiček. Proto by nebylo vhodné převzít všemi „hobbyisty" a amatéry uznávanou sběrnici S 100 (sto špiček) ani sběrnici MULTIBUS (86 špiček). Z diskuse vyplynulo kompromisní řešení vycházející z evropské sběrnice MU-BUS, která je na počet špiček konektorů méně náročná. Ale i tato sběrnice musela být upravena.

Dalším z řešených problémů je jednořádkový alfanumerický displej s kurzorem na TVP, vyvinutý v VD ČSAV, jehož konstrukce měla být původně součástí tohoto čísla (vyjde v AR řady A). V současné době je řešen šestnáctiřádkový displej.

Vzhledem k nedostatku polovodičových pamětí řeší jeden ze žáků ZDŠ použití feritové paměti (4k byte) z počítače ZUSE 25 pro osobní počítač INTELKA 80.

Zároveň probíhají zkoušky jednoduchých periferních zařízení (jako jsou jednoduché snímače osmistopé děrné pásky a ověřuje se laciná alfanumerická klávesnice z knoflíků na peřiny).

V současné době jsou i vyhodnocovány zkušenosti ze zápisu programů a dat na kazetový magnetofon. Ve zkouškách byl ověřován systém převzatý z MCS85 a systém, který používá modulovaného záznamu ze sériového výstupu UART při rychlosti 200 Bd.

I když se většina ze zájemců o stavbu osobního počítače INTELKA 80 rozhodla pro mikroprocesor 8080, přesto se našlo několik jedinců, kteří se rozhodli pro mikroprocesor Z80 a nebo I8085.

Diskuse o využití osobního počítače INTELKA 80 vedou k dalším návrhům interface desek pro tento počítač. Po dokončení základní sestavy máme v plánu zhotovit desky pro grafický displej, ALU s kalkulačkovými čipy z TI 58, skupinový vyučovací systém, individuální vyučovací systém a dále programové vybavení pro nejrůznější hry a použití osobního počítače INTELKA 80 při registraci a řízení zájmové činnosti, skladovém hospodářství, administrativě, v amatérské vysílací a přijímací technice, v domácnosti při řízení topení, spotřeby energie apod.

Předpokládáme, že vás na stránkách Amatérského radia budeme moci o stavbě osobního počítače INTELKA 80 informovat co nejdříve.